TESIS - KI142502

PERBAIKAN PREDIKSI KESALAHAN PERANGKAT LUNAK MENGGUNAKAN SELEKSI FITUR DAN *CLUSTER-BASED CLASSIFICATION*

FACHRUL PRALIENKA BANI MUHAMAD
5115201044

DOSEN PEMBIMBING I
Daniel Oranova Siahaan, S.Kom., M.Sc, PD.Eng.
NIP. 197411232006041001

DOSEN PEMBIMBING II
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
NIP. 197512202001122002

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

THESIS - KI142502

IMPROVEMENT OF SOFTWARE FAULT PREDICTION USING FEATURE SELECTION AND CLUSTER-BASED CLASSIFICATION

FACHRUL PRALIENKA BANI MUHAMAD
5115201044

SUPERVISOR I
Daniel Oranova Siahaan, S.Kom., M.Sc, PD.Eng.
NIP. 197411232006041001

SUPERVISOR II
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
NIP. 197512202001122002

MASTER PROGRAM
DEPARTEMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
FACHRUL PRALIENKA BANI MUHAMAD
NRP. 5115201044

Dengan judul:
PERBAIKAN PREDIKSI KESALAHAN PERANGKAT LUNAK MENGGUNAKAN SELEKSI FITUR DAN *CLUSTER-BASED CLASSIFICATION*

Tanggal Ujian: 11 Juli 2017
Periode Wisuda: 2016 Genap

Disetujui oleh:

Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng. (Pembimbing 1)
NIP. 197411232006041001

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. (Pembimbing 2)
NIP. 197512202001122002

Prof. Drs. Ec. Ir. Riyanarto S., M.Sc., Ph.D. (Penguji 1)
NIP. 195908031986011001

Sarwosri, S.Kom., MT. (Penguji 2)
NIP. 197608092001122001

Adhatus Solichah Ahmadiyah S.Kom, M.Sc (Penguji 3)
NIP. 198508262015042002

Dekan Fakultas Teknologi Informasi,

Dr. Agus Zainal Arifin, S.Kom., M.Kom.
NIP. 197208091995121001

[Halaman ini sengaja dikosongkan]

PERBAIKAN PREDIKSI KESALAHAN PERANGKAT LUNAK MENGGUNAKAN SELEKSI FITUR DAN *CLUSTER-BASED CLASSIFICATION*

Nama mahasiswa	:	Fachrul Pralienka Bani Muhamad
NRP	:	5115201044
Pembimbing	:	Daniel Oranova Siahaan, S.Kom., M.Sc, PD.Eng. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRAK

Nilai *balance* pada hasil prediksi kesalahan perangkat lunak perlu menjadi perhatian, karena pada umumnya persebaran label kelas *true* dan *false* pada dataset kesalahan perangkat lunak cenderung tidak seimbang. Nilai *balance* diperoleh dari hasil tarik-ulur *(trade-off)* antara nilai *probability detection (pd)* dan *probability false alarm (pf)*. Peneliti sebelumnya telah mengusulkan metode untuk memprediksi kesalahan perangkat lunak, yaitu *cluster-based classification (CBC)* yang diintegrasikan dengan *entropy-based discretization (EBD)* di tujuh dataset *NASA public MDP*. Hasil penelitian sebelumnya menunjukkan nilai rata-rata *balance* adalah 67,65% dengan 83,3% *pd* dan 40,3% *pf*.

Penelitian ini mengusulkan perbaikan nilai *balance* dari hasil prediksi kesalahan perangkat lunak pada metode *CBC*, dengan mengintegrasikan metode seleksi fitur. Adapun metode seleksi fitur yang akan diintegrasikan dengan *CBC* yaitu *gain ratio (GR)*, *information gain (IG)*, *one-r (OR)*, *relief-f (RFF)*, dan *symmetric uncertainty (SU)*. Hal ini didasarkan pada penelitian sebelumnya, dimana model prediksi dengan fitur yang redundan dan tidak relevan pada dataset kesalahan perangkat lunak dapat menurunkan nilai *pd* dan meningkatkan nilai *pf*, sehingga nilai *balance* cenderung menurun.

Berdasarkan hasil penelitian dengan menggunakan tujuh dataset *NASA public MDP*, ditunjukkan bahwa *CBC* yang dikombinasikan dengan metode seleksi fitur *IG* dapat menghasilkan nilai rata-rata *balance* paling tinggi jika dibandingkan dengan rata-rata nilai *balance* di metode seleksi fitur lainnya, yaitu 63,73%. Jika dibandingkan dengan metode *CBC* pada penelitian sebelumnya di lima dataset yang sama, maka usulan kombinasi metode *CBC* dengan seleksi fitur *IG* dapat menghasilkan nilai rata-rata *balance* 0,52% yang lebih rendah, yaitu 63,91% pada usulan metode dan 64,43% pada penelitian sebelumnya. Namun, metode yang diusulkan dapat menghasilkan nilai *balance* yang lebih baik pada 3 dari 5 dataset, yaitu CM1 (69,05%), MW1 (62,90%), dan PC4 (72,70%).

Kata kunci: *Cluster-based classification*, *Entropy-based discretization*, Kesalahan perangkat lunak, *NASA public MDP*, Seleksi Fitur.

[Halaman ini sengaja dikosongkan]

IMPROVEMENT OF SOFTWARE FAULT PREDICTION USING FEATURE SELECTION AND CLUSTER-BASED CLASSIFICATION

Name	: Fachrul Pralienka Bani Muhamad
Student Identity Number	: 5115201044
Supervisor	: Daniel Oranova Siahaan, S.Kom., M.Sc, PD.Eng. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRACT

The balance values in software fault prediction results should be of concern, since in most cases the distribution of true and false label labels on the software fault dataset tends to be unbalanced. The balance value is obtained from the trade-off between probability detection (pd) and probability false alarm (pf). Previous researchers have proposed a method for predicting software errors, namely cluster-based classification (CBC) which is integrated with entropy-based discretization (EBD) in 7 NASA public MDP datasets. The results of the previous study showed that the average balance value was 67.65% with 83.3% pd and 40.3% pf.

This study proposes an improvement of balance value in software fault prediction results using CBC, by integrating feature selection methods. The feature selection methods that will be integrated on CBC are gain ratio (GR), information gain (IG), one-r (OR), relief-f (RFF), and symmetric uncertainty (SU). Based on the previous research, predictive models with redundant and irrelevant features in the software fault dataset can decrease the pd value and increase the pf value, so that the balance value tends to decrease.

Based on the research using 7 NASA public MDP datasets, it was shown that CBC which combined with IG can yield the highest average value of balance when compared to other feature selection method in this research, i.e. 63.73% balance. When it compared to CBC in previous study using the same 5 NASA datasets, the proposed combination can result in 0.52% lower balance average, i.e. 63,91% on proposed method and 64,43% on previous research. However, the proposed method can yield better balance values in 3 of 5 datasets, i.e. CM1 (69.05%), MW1 (62.90%), and PC4 (72.70%).

Keywords: Cluster-based classification, Entropy-based discretization, Feature selsction, NASA public MDP, Software fault prediction.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis bisa menyelesaikan Tesis yang berjudul "Perbaikan Prediksi Kesalahan Perangkat Lunak Menggunakan Seleksi Fitur Dan *Cluster-Based Classification*" sesuai dengan target waktu yang diharapkan. Pengerjaan Tesis ini adalah suatu kesempatan yang sangat berharga bagi penulis untuk belajar memperdalam ilmu pengetahuan. Terselesaikannya buku Tesis ini tidak terlepas dari bantuan dan dukungan semua pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat-Nya sehingga penulis dapat menyelesaikan Tesis ini dengan baik.
2. Kedua orang tua penulis yang selalu mendoakan agar selalu diberikan kelancaran dan kemudahan dalam menyelesaikan Tesis ini. Serta menjadi motivasi terbesar untuk mendapatkan hasil yang terbaik.
3. Bapak Daniel Oranova Siahaan, S.Kom., M.Sc, PD.Eng. dan Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku dosen pembimbing yang telah memberikan kepercayaan, motivasi, bimbingan, nasehat, perhatian serta semua bantuan yang telah diberikan kepada penulis dalam menyelesaikan buku tesis ini.
4. Bapak Prof. Drs. Ec. Ir. Riyanarto S., M.Sc., Ph.D., Ibu Sarwosri, S.Kom., MT., dan Ibu Adhatus Solichah Ahmadiyah S.Kom, M.Sc selaku dosen penguji yang telah memberikan bimbingan, saran, arahan, dan koreksi dalam pengerjaan buku tesis ini.
5. Bapak Waskitho Wibisono, S.Kom., M.Eng., PhD selaku ketua program pascasarjana Teknik Informatika ITS, dan Bapak Dr. Ir. R.V. Hari Ginardi, M.Kom. selaku dosen wali penulis dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.

7. Seluruh keluarga besar yang selalu memberi semangat, doa, dukungan kepada penulis.
8. Rekan-rekan mahasiswa Pasca Sarjana Teknik Informatika ITS tahun 2015 yang telah menemani dan memberikan bantuan serta motivasi untuk segera menyelesaikan buku tesis ini.
9. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penyelesaian buku tesis ini.

Sebagai manusia biasa, penulis menyadari bahwa buku tesis ini masih jauh dari kesempurnaan dan terdapat banyak kekurangan. Sehingga dengan segala kerendahan hati, penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, 2017

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Permintaan sistem yang andal dan aman selalu meningkat dari waktu ke waktu. Tingginya tingkat keandalan dan keamanan dapat membuat pengembangan sistem menjadi lebih kompleks. Peningkatan kompleksitas sistem mempengaruhi tingkat ketelitian yang dibutuhkan pada tahap pengujian, sehingga dapat sangat rentan terhadap kesalahan dan sangat berpotensi terhadap bahaya. Kesalahan perangkat lunak menjadi penyebab utama terjadinya kegagalan perangkat lunak. Kegagalan pada perangkat lunak dapat meningkatkan jumlah biaya dan sumber daya yang digunakan untuk proses pengujian pada tahap selanjutnya. Selain itu, kegagalan pada sistem kritis, sistem keselamatan, dan sistem bisnis, dapat berdampak pada kehidupan dan keselamatan manusia (Akalya Devi et al. 2012). Kegagalan yang terjadi pada perangkat lunak menandakan rendahnya kualitas perangkat lunak tersebut.

Kualitas perangkat lunak yang tinggi memerlukan kecermatan pada perencanaan biaya dan manajemen sumber daya pengujian. Kualitas perangkat lunak menjadi salah satu area penelitian di bidang rekayasa perangkat lunak yang memiliki peranan penting dalam menjamin kualitas dari pengembangan sebuah sistem. Beberapa poin yang dijadikan sebagai jaminan kualitas perangkat lunak yaitu keandalan sistem, kerentanan sistem terhadap kesalahan, dan kemampuan sistem untuk dapat digunakan kembali (Akalya Devi et al. 2012). Rekayasa kualitas perangkat lunak terdiri dari beberapa kegiatan penjamin kualitas yang meliputi pengujian, verifikasi, inspeksi, toleransi kesalahan, dan prediksi kesalahan (Catal & Diri 2009).

Pada sebuah siklus hidup pengembangan perangkat lunak, tahap pengujian adalah tahap yang paling banyak mengkonsumsi waktu dan sumber daya. Sekitar 50% jadwal keseluruhan proyek digunakan untuk menguji perangkat lunak (Singh & Verma 2014). Sebuah kesalahan perangkat lunak dapat selalu diamati saat proses pengujian berlangsung. Pada kegiatan prediksi kesalahan perangkat lunak, setiap kesalahan yang terjadi pada proses pengujian, ditandai sebagai data kesalahan

perangkat lunak. Matrik perangkat lunak yang berisi pengukuran dari beberapa properti perangkat lunak dan data-data kesalahan perangkat lunak tersebut akan digunakan untuk mengidentifikasi modul yang paling mungkin terdapat potensi kesalahan di tahap rilis selanjutnya.

Prediksi kesalahan perangkat lunak sangat penting untuk dilakukan, karena alokasi biaya dan sumber daya pengujian terbatas. Selain itu, dengan prediksi kesalahan perangkat lunak, proses pengujian dapat difokuskan pada modul-modul yang rawan terhadap kesalahan, sehingga sumber daya pengujian dapat dihemat untuk modul lain yang rentan salah. Kegiatan prediksi kesalahan perangkat lunak adalah kegiatan penjamin kualitas yang paling efisien, karena dapat mengurangi waktu pengujian dengan menghemat sumber daya pengujian.

Model prediksi kesalahan perangkat lunak dapat digunakan untuk mengelola sumber daya pengujian secara efisien, namun belum tentu efektif meningkatkan kualitas perangkat lunak. Salah satu faktor penyebabnya adalah rendahnya nilai *probability detection (pd)* dan tingginya nilai *probability false alarm (pf)*. Nilai *pd* adalah nilai keberhasilan sistem dalam memprediksi kesalahan perangkat lunak. Sedangkan nilai yang seharusnya bertolak belakang dengan *pd* adalah nilai *pf*. Nilai *pf* adalah nilai misklasifikasi sistem dalam menentukan modul yang secara aktual tidak terdapat kesalahan namun diklasifikasikan sebagai modul yang salah. Secara ideal, sistem prediksi kesalahan perangkat lunak yang baik dapat menghasilkan nilai *pd* setinggi mungkin dengan nilai *pf* yang serendah mungkin. Sedangkan untuk menentukan nilai *pd* dan *pf* yang terbaik diperlukan nilai *balance*. Nilai *balance* adalah nilai tarik-ulur *(trade-off)* antara nilai *pd* dan nilai *pf*. Menurut penelitian sebelumnya, teknik prediksi kesalahan tidak hanya dinilai berdasarkan performa kinerjanya saja, tetapi juga aspek lainnya yaitu efisiensi komputasi dan efektifitas yang dihasilkan oleh teknik prediksi (Moeyersoms et al. 2015).

Para peneliti sebelumnya telah mengusulkan beberapa metode data *mining* untuk melakukan prediksi kesalahan perangkat lunak. Metode-metode yang diusulkan tersebut antara lain *Genetic Programming*, *Decision Tree*, *Neural Network*, *Density-based Clustering*, *Case-based Reasoning*, *Fuzzy Logic*, *Logistic Regression*, dan *Naïve Bayes* (Singh & Verma 2014). Adapun dataset yang umumnya digunakan para peneliti untuk memprediksi kesalahan perangkat lunak

adalah *NASA public MDP (Modular toolkit for Data Processing)* yang berisi sejumlah data numerik dari matrik perangkat lunak, misalnya jumlah *line of code*, kompleksitas *McCabe/Halstead*, dan label data kesalahan perangkat lunak (Najadat & Alsmadi 2012). Berdasarkan hasil penelitian, secara umum metode yang dapat menghasilkan tingkat akurasi yang tinggi cenderung menggunakan teknik pemodelan yang sederhana, yaitu *naïve bayes* (Hall et al. 2012). Metode *Naïve Bayes* dapat menghasilkan sekitar 71% *pd* dan 25% *pf* pada dataset *NASA public MDP* (Menzies et al. 2007). Namun, menurut penelitian yang dilakukan Predeep Singh dan Shrish Verma, jika dibandingkan dengan metode *Naive Bayes*, maka metode *cluster-based classification (CBC)* dapat menghasilkan nilai *pd* yang lebih baik, yaitu 83,3% *pd* dengan 40,3% *pf* (Singh & Verma 2014). Perbaikan hasil prediksi kesalahan perangkat lunak yang diusulkan Predeep Singh dan Shrish Verma tidak hanya ditentukan oleh metode prediksi *CBC* saja, tetapi juga dengan tahap pra-proses diskritisasi data numerik yaitu *Entropy-based Discretization (EBD)*.

Diskritisasi data numerik pada dataset *NASA public MDP* dilakukan agar data dapat lebih mendekati tingkat representasi pengetahuan. Jika dibandingkan dengan data kontinyu (numerik), maka data diskrit lebih stabil dan akurat terhadap algortima prediksi kesalahan perangkat lunak. Gagasan utama dalam melakukan diskritisasi data numerik pada dataset *NASA public MDP* adalah untuk meningkatkan efisiensi komputasi, karena jumlah nilai pada atribut kontinyu dikurangi dengan membaginya ke dalam beberapa interval (Singh & Verma 2009).

Selain diskritisasi data, beberapa penelitian memaparkan bahwa seleksi fitur juga dapat meningkatkan akurasi pada prediksi kesalahan perangkat lunak. Pengembangan metode prediksi dengan fitur yang tidak relevan dan berlebihan dari data kesalahan perangkat lunak dapat membuat akurasi prediksi menjadi lebih rendah. Seleksi fitur terbagi menjadi tiga kategori, yaitu *filter*, *wrapper*, dan *embedded*. Berdasarkan penelitian sebelumnya (Antony et al. 2016), lima metode seleksi fitur yang termasuk ke dalam kategori seleksi fitur *filter* diusulkan untuk meningkatkan nilai *pd* dari hasil prediksi kesalahan perangkat lunak. Kelima metode seleksi fitur tersebut adalah *Gain Ratio (GR), Information Gain (IG), One-R (OR), Relief-F (RFF),* dan *Symmteric Uncertainty (SU)*. Metode seleksi fitur

dikombinasikan dengan metode prediksi *Naive Bayes*. Kombinasi metode *Naive Bayes* dengan *One-R* menunjukkan hasil *pd* yang paling tinggi dibandingkan yang lain (Singh et al. 2016).

Berdasarkan hasil dari beberapa penelitian sebelumnya, penggunaan semua fitur pada dataset kesalahan perangkat lunak dapat membuat nilai prediksi menurun. Hal itu disebabkan oleh redundansi fitur dan irrelevansi fitur terhadap label *class* kesalahan perangkat lunak. Selain itu, nilai *pf* cenderung meningkat seiring dengan meningkatnya nilai *pd*. Salah satu faktor penyebabnya adalah tidak seimbangnya persebaran label kelas pada dataset kesalahan perangkat lunak (Singh & Verma 2014). Hasil yang ideal pada prediksi kesalahan perangkat lunak yaitu nilai *pd* mendekati 100% dengan nilai *pf* mendekati 0% (Menzies et al. 2007). Kondisi tersebut dapat dicari dengan memperhatikan nilai *balance*.

Oleh karena itu, pada penelitian ini diusulkan perbaikan nilai *balance* pada kombinasi metode *EBD* dan *CBC* dengan menggunakan lima metode seleksi fitur. Dari kelima metode seleksi fitur tersebut, akan dilakukan analisis untuk menemukan kombinasi metode seleksi fitur mana yang dapat menghasilkan nilai *pd* dan *pf* dengan *balance* yang paling tinggi pada *CBC*. Namun, dikarenakan tingginya interval data pada setiap dataset kesalahan perangkat lunak, jika dilakukan diskritisasi, maka besar kemungkinan ditemukan duplikasi data diskrit dengan kelas yang berbeda. Duplikasi data dengan kelas yang berbeda tersebut akan membuat model prediksi menjadi tidak konsisten. Oleh karena itu, selain kombinasi seleksi fitur, diusulkan pula penanganan data redundan dengan kelas yang berbeda yaitu menggunakan split *EBD* dua fase. Hasil kombinasi tersebut diharapkan dapat meningkatkan nilai *balance* pada hasil prediksi kesalahan perangkat lunak.

Adapun susunan pada penelitian ini yaitu, BAB 2 dijelaskan tentang ide umum dari metode-metode yang digunakan dalam penelitian ini. BAB 3 berisi pemaparan rancangan usulan metode yang akan dilakukan untuk memprediksi kesalahan perangkat lunak melalui beberapa kombinasi metode, beserta langkah-langkah untuk melakukan validasi terhadap usulan kombinasi metode tersebut. BAB 4 penyajian hasil model prediksi kesalahan perangkat lunak menggunakan kombinasi *CBC* dengan lima metode seleksi fitur *(IG, GR, OR, RFF, SU)*. Selain itu pada BAB 4 disajikan pula skenario dan uji coba model prediksi kesalahan

perangkat lunak. Sedangkan BAB 5 berisi kesimpulan dan saran dari hasil percobaan yang telah dilakukan.

1.2. Perumusan Masalah

Rumusan masalah dalam penelitian ini yaitu:

1. Bagaimana mengembangkan metode prediksi kesalahan perangkat lunak pada *CBC* dengan menggunakan metode seleksi fitur?
2. Bagaimana menentukan metode seleksi fitur yang dapat meningkatkan nilai *balance* pada metode *CBC*?
3. Bagaimana menentukan jumlah fitur terbaik yang digunakan sebagai *input* model prediksi kesalahan perangkat lunak menggunakan *CBC*?

1.3. Batasan Masalah

Masalah pada area penelitian ini dibatasi sebagai berikut:

1. Penelitian yang dilakukan berfokus pada nilai *balance* dari hasil prediksi kesalahan perangkat lunak.
2. Dataset yang digunakan adalah tujuh dataset kesalahan perangkat lunak dari *NASA public MDP* yaitu CM1, KC3, MW1, PC1, PC2, PC3, dan PC4.

1.4. Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah meningkatkan nilai *balance* pada prediksi kesalahan perangkat lunak dengan cara mengkombinasikan lima metode seleksi fitur dengan metode prediksi *CBC*. Usulan metode tersebut diharapkan dapat digunakan untuk mengelola sumber daya pengujian secara efisien dan efektif.

1.5. Kontribusi Penelitian

Kontribusi pada penelitian ini adalah usulan kombinasi metode *CBC* dengan lima metode seleksi fitur untuk meningkatkan nilai *balance* pada prediksi kesalahan perangkat lunak. Selain itu, diusulkan pula metode tambahan untuk mengatasi hasil diskritisasi yang redundan dengan kelas yang berbeda, yaitu split *EBD* dua fase.

[Halaman ini sengaja dikosongkan]

BAB 2
KAJIAN PUSTAKA

Pada bab ini dijelaskan tentang penelitian terkait pada prediksi kesalahan perangkat lunak. Selain itu, dipaparkan juga pengertian kesalahan perangkat lunak, dataset *NASA public MDP*, dan dasar teori pendukung pada bab tiga. Adapun dasar teori yang digunakan pada bab tiga yaitu *entropy-based discretization (EBD), cluster-based discretization (CBC), information gain (IG), gain ratio (GR), one-r (OR), relief-f (RFF), symmetric uncertainty (SU),* nilai *pd*, nilai *pf*, dan nilai *balance*.

2.1. Penelitian Terkait

Beberapa penelitian terkait dengan topik prediksi kesalahan perangkat lunak di antaranya penelitian yang dilakukan oleh (Hall et al. 2012), (Malhotra 2015), (Arora et al. 2015), dan (Catal 2011) yaitu tentang *systematic literature review* untuk menemukan *trend* solusi pada prediksi kesalahan perangkat lunak. Hasil penelitian menunjukkan bahwa metode statistik sebagai solusi prediksi kesalahan perangkat lunak sudah mulai ditinggalkan, sedangkan area penelitian menggunakan data *mining* untuk memprediksi kesalahan adalah yang paling popular. Hal tersebut dikarenakan banyaknya ragam variasi metode dalam data *mining* untuk meningkatkan nilai akurasi prediksi kesalahan perangkat lunak. Metode data *mining* yang dapat menghasilkan tingkat akurasi prediksi yang tinggi cenderung menggunakan metode yang sederhana, misalnya *naïve bayes*.

(Menzies et al. 2007) mengusulkan metode *naïve bayes* untuk memprediksi kesalahan perangkat lunak. Usulan metode tersebut diterapkan pada 8 dataset *NASA public MDP* dengan melakukan pra-proses *log number* yang dikombinasikan dengan metode seleksi fitur *IG*. Hasil penelitian menunjukkan nilai 71% *pd* dan 25% *pf*.

(Kumar & Zhang 2007) membuktikan bahwa hasil klasifikasi *k-NN* dengan tahapan diskritisasi data numerik menggunakan *EBD* pada pra-proses dapat meningkatkan akurasi prediksi. Hal tersebut didukung dengan dataset *hand geometri* yang berisi data numerik. Selain itu, metode *EBD* dapat mengurangi

kemungkinan *over-fitting* pada model prediksi dan memproses prediksi dengan lebih cepat dibandingkan dengan data kontinyu (numerik) yang tidak didiskritisasi.

(Singh & Verma 2009) telah mencoba kombinasi *naïve bayes* dengan pra-proses *EBD* pada dua dataset kesalahan *embedded software system* yang didapat dari *NASA public MDP*. Hasil menunjukkan bahwa *EBD* dapat membantu meningkatkan akurasi *naive bayes* dalam memprediksi kesalahan perangkat lunak. Penelitian ini juga membuktikan bahwa karakteristik beberapa dataset *embedded software system* pada *NASA public MDP* cocok untuk dilakukan diskritisasi *EBD.*

(Singh & Vyas 2014) dan (Singh & Verma 2014) mengusulkan metode *CBC* untuk memprediksi kesalahan perangkat lunak. Penelitian yang pertama mengkombinasikan *CBC* dengan *EBD* pada tiga dataset kesalahan perangkat lunak yang didapat dari *NASA public MDP*. Hasil menunjukkan bahwa akurasi metode *CBC* dengan *EBD* lebih unggul dari metode *ANN*. Sedangkan penelitian yang kedua melakukan hal yang sama dengan penelitian pertama, tetapi dengan dataset yang digunakan lebih beragam yaitu 7 dataset kesalahan perangkat lunak dari *NASA public MDP*. Hasil penelitian menunjukkan bahwa *CBC* dan *EBD* menghasilkan tingkat akurasi yang lebih tinggi jika dibandingkan dengan *naïve bayes* yang sebelumnya cenderung menghasilkan nilai akurasi yang tinggi. Nilai *pd* meningkat menjadi 83% dan nilai *pf* menjadi 40%.

(Singh et al. 2016) menggunakan metode seleksi fitur untuk meningkatkan hasil akurasi prediksi pada *naïve bayes*. Penelitian tersebut dilakukan pada dataset *Turkish software industry*. Hasil penelitian menunjukkan bahwa metode seleksi fitur *OR* menghasilkan nilai *pd* terbaik dibandingkan keempat metode seleksi fitur lainnya yaitu *IG, GR, RFF,* dan *SU*.

(Akbar 2017) mengusulkan kombinasi metode *naïve bayes* dengan *GR*. Hasil penelitian menunjukkan bahwa akurasi prediksi kesalahan perangkat lunak pada dataset *NASA public MDP* dapat ditingkatkan dengan seleksi fitur *GR*. Untuk mendukung metode *naïve bayes* yang kooperatif pada data diskrit, maka data kesalahan perangkat lunak didiskritisasi terlebih dahulu ke dalam lima kategori secara acak.

(Novakovic 2010) meneliti pengaruh seleksi fitur terhadap model prediksi *naïve bayes*. Peneliti tersebut menggunakan lima dataset yang diperoleh dari *UCI*

repository yaitu *mushroom*, *vote*, *credit*, *audiology*, dan *M2*. Beberapa metode seleksi fitur yang diintegrasikan dengan *naïve bayes* adalah *IG*, *GR*, *SU*, *OR*, *RFF*, dan *chi-squared (CS)*. Hasil menunjukkan bahwa *OR* adalah metode seleksi fitur yang paling tinggi dalam meningkatkan akurasi metode *naïve bayes* dalam melakukan klasifikasi.

Tabel 2.1 Perbandingan Metode Pada Penelitian Terkait

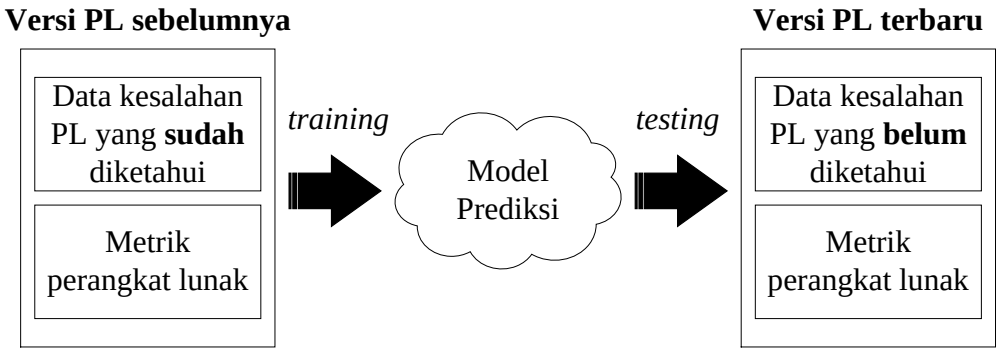
No	Metode		Hasil			Dataset
	Pra-proses	Proses	Akurasi	*pd (%)*	*pf (%)*	
1	*LogNum* dan IG	*NB*	-	71	25	8 dataset *NASA public MDP* (CM1, KC3, KC4, MW1, PC1, PC2, PC3, dan PC4)
2	*EBD*	*CBC*	-	83	40	7 dataset *NASA public MDP* (CM1, KC3, MW1, PC1, PC2, PC3, dan PC4)
3	*EBD*	*K-NN*	95	-	-	Dataset *hand-geometri*
4	*EBD*	*NB*		76	9,2	Dataset *embed-software system* (AR3, AR4, AR5)
5	*OR*	*NB*	87,4	-	-	10 dataset kesalahan perangkat lunak (AR1, AR3, AR4, AR5, AR6, KC1, KC2, KC3, CM1, Datatrieve)
6	*GR*	*NB*	-	84,86	-	5 dataset *NASA public MDP* (CM1, JM1, KC1, KC2, PC1)
7	-	NB	95,8	-	-	Dataset *mushroom*
8	*IG*	*NB*	97,1	-	-	Dataset *mushroom*

Tabel 2.1 adalah rangkuman metode-metode penelitian terkait yang telah dijelaskan sebelumnya. Berdasarkan Tabel 2.1, jumlah kesamaan penggunaan dataset yang paling banyak ada di metode nomor 1 dan nomor 2, yaitu dataset *NASA public MDP*. Jika pada penelitian nomor 1 menggunakan 7 dataset yang sama dengan penelitian nomor 2, maka metode nomor 1 dapat menghasilkan 70% *pd* dan 23% *pf*. Hal ini membuktikan bahwa metode nomor 2 dapat menghasilkan 13% *pd* dan 19% *pf* yang lebih tinggi. Dikarenakan adanya kenaikan nilai *pd* pada dataset *NASA public MDP*, maka metode nomor 2 akan digunakan sebagai metode acuan dalam penelitian ini *(EBD & CBC)*.

Jika akurasi *NB* di metode nomor 7 dibandingkan dengan akurasi *NB* yang telah diintegrasikan dengan seleksi fitur *IG* di metode nomor 8, maka terdapat peningkatan akurasi sebesar 1,3%. Hal ini membuktikan bahwa metode seleksi fitur *IG* dapat meningkatkan akurasi *NB* sebesar 1,3%. Berdasarkan penelitian tersebut, maka *EBD* dan *CBC* akan diintegrasikan dengan metode seleksi fitur. Namun, pada penelitian ini metode seleksi fitur yang digunakan tidak hanya *IG* saja, tetapi juga 4 metode seleksi fitur lainnya yaitu *GR*, OR, *RFF*, dan *SU*. Dengan seleksi fitur, nilai *balance* metode *EBD* dan *CBC* diharapkan akan meningkat lebih dari 1,3%.

2.2. Kesalahan Perangkat Lunak

Kesalahan perangkat lunak adalah masalah utama dalam sistem perangkat lunak yang perlu diminimalisasi untuk memastikan keandalannya (Erturk & Sezer 2015). Setiap kesalahan perangkat lunak dapat selalu diamati pada saat proses pengujian berlangsung. Kesalahan perangkat lunak dapat menyebabkan kegagalan pada perangkat lunak sehingga berdampak pada peningkatan biaya dan usaha untuk memperbaiki kesalahan tersebut. Kegagalan perangkat lunak menandakan rendahnya tingkat kualitas dari perangkat lunak. Untuk meningkatkan tingkat kualitas perangkat lunak, maka kesalahan perangkat lunak perlu untuk diminimalisasi. Prediksi kesalahan perangkat lunak adalah kegiatan memprediksi modul yang rentan terhadap kesalahan di tahap rilis selanjutnya dengan menggunakan matrik prediksi dan data kesalahan perangkat lunak pada versi sebelumnya. Secara umum, proses prediksi kesalahan perangkat lunak dapat dilihat pada Gambar 2.1.



Gambar 2.1 Proses Prediksi Kesalahan Perangkat Lunak

Adapun perbedaan dari *error*, kegagalan, kesalahan, dan cacat perangkat lunak adalah sebagai berikut (Erturk & Sezer 2015):

1. *Error* adalah tindakan tertentu manusia yang seharusnya dilakukan dari proses suatu produk namun terlewati atau tidak dilakukan. Misalnya, *error* yang dibuat ketika melakukan implementasi kode program.
2. Kegagalan perangkat lunak terjadi ketika perilaku perangkat lunak tidak memenuhi harapan pengguna.
3. Kesalahan perangkat lunak adalah langkah yang salah pada kode program, proses, pendefinisian data, atau cacat fisik yang terjadi pada perangkat keras. Misalnya, perangkat lunak dapat masuk ke dalam keadaan yang tidak

diinginkan karena kesalahan *programmer* sehingga menyebabkan kegagalan perangkat lunak.

4. Cacat perangkat lunak adalah istilah umum yang digunakan untuk menyebut *error*, kesalahan, dan kegagalan. Jadi, dapat disimpulkan bahwa *error* dapat menyebabkan kesalahan, lalu kesalahan perangkat lunak dapat menyebabkan kegagalan ketika perangkat lunak sudah berjalan, dan istilah cacat perangkat lunak mencakup semua pengertian-pengertian sebelumnya.

2.3. Dataset *NASA public MDP*

Prediksi kesalahan perangkat lunak adalah salah satu faktor penting dalam meningkatkan kualitas pada proses pengembangan perangkat lunak. Seperti yang sudah dijelaskan sebelumnya, bahwa untuk memprediksi kesalahan perangkat lunak diperlukan beberapa matrik dan data kesalahan perangkat lunak dari versi perangkat lunak sebelumnya. Matrik perangkat lunak adalah ukuran kuantitatif sederhana dari setiap atribut pada siklus hidup perangkat lunak. Matrik perangkat lunak memungkinkan bagi peneliti untuk mengukur dan memprediksi proses perangkat lunak, sumber daya yang diperlukan, dan produk kerja yang relevan untuk usaha pengembangan perangkat lunak. Matrik perangkat lunak yang sering digunakan para peneliti untuk mengukur kompleksitas kode program yaitu *line of code (loc),* kompleksitas *mcCabe*, dan kompleksitas *halstead* (Singh & Verma 2009).

Berbagai dataset yang berisi variasi matrik telah digunakan untuk memprediksi kesalahan perangkat lunak, dan beberapa di antaranya bersifat *private*. Dikarenakan tidak adanya akses untuk menggunakan dataset *private*, maka dalam penelitian ini akan memanfaatkan dataset *public* yang dapat diakses oleh semua peneliti dan dapat digunakan untuk berbagai keperluan. Salah satu dataset kesalahan perangkat lunak yang telah banyak diteliti adalah *NASA public MDP*. Dataset tersebut dapat diakases secara umum melalui repositori dataset rekayasa perangkat lunak *promise*. Dataset *NASA public MDP* umumnya juga berisi data numerik yang terdiri dari *line of code*, kompleksitas *mcCabe*, kompleksitas *halstead*, dan label data kesalahan perangkat lunak (Lessmann et al. 2008). Sebanyak tujuh dataset akan digunakan pada penelitian ini yaitu CM1, KC3, MW1,

PC1, PC2, PC3, dan PC4. Jumlah dan pilihan dataset disesuaikan dengan penelitian sebelumnya (Singh & Verma 2014). Pada setiap dataset terdapat jumlah modul perangkat lunak yang berbeda-beda. Modul perangkat lunak adalah unit terkecil dari fungsi dalam sistem secara keseluruhan (Najadat & Alsmadi 2012). Adapun rincian setiap dataset ada di Tabel 2.2 (Singh & Verma 2015) (Sathyaraj & Prabu 2015).

Tabel 2.2 Rincian 7 Dataset *NASA public MDP*

No	Nama	Modul	*Fault*	*%*	Bahasa	Deskripsi
1	CM1	498	49	9,84	C/C++	Instrumen pesawat ruang angkasa NASA
2	KC3	458	43	9,38	Java	Manajemen menyimpanan untuk data ground
3	MW1	403	31	7,69	C	Percobaan gravitasi nol terhadap pembakaran
4	PC1	1109	77	6,94	C	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi
5	PC2	5589	23	0,41	C	Simulator dinamis pada sistem control
6	PC3	1563	160	10,24	C	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi
7	PC4	1458	178	12,21	C	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi

2.4. *Entropy-based Discretization (EBD)*

Metode data *mining* telah berhasil diterapkan untuk menyelesaikan masalah prediksi atau klasifikasi pada berbagai latar belakang permasalahan. Dalam data *mining*, proses diskritisasi dikenal sebagai salah satu kegiatan pra-proses data yang sangat penting. Sebagian besar algoritma pada metode data *mining* mampu mengekstraksi pengetahuan dari dataset dengan fitur yang berisi data diskrit. Jika data pada fitur dataset adalah kontinyu (numerik), maka metode prediksi dapat diintegrasikan dengan algoritma diskritisasi yang mengubah fitur numerik menjadi fitur diskrit.

Metode diskritisasi digunakan untuk mengurangi jumlah nilai pada fitur kontinyu (numerik) dengan membagi jarak setiap atribut ke dalam interval tertentu (Singh & Verma 2009). Label interval tersebut dapat digunakan untuk mengganti nilai data aktual. Diskritisasi membuat proses data *mining* menjadi lebih cepat dan akurat. Secara umum, proses diskritisasi ada empat langkah yaitu mengurutkan semua nilai numerik yang akan di-diskritisasi, kemudian pilih titik *splitting* pada nilai numerik tersebut, lalu *split* atau gabungkan kedua titik nilai numerik, dan yang terakhir pilih kriteria *stopping* pada proses diskritisasi.

Proses diskritisasi data pada permasalahan klasifikasi umumnya dibagi menjadi dua yaitu *supervised* dan *unsupervised*. Metode diskritisasi *supervised* dilakukan ketika terdapat interdependensi pengetahuan antara label kelas dan data pada fitur. Sedangkan diskritisasi *unsupervised* dilakukan ketika tidak adanya atau tidak digunakannya label kelas dari setiap data. Dikarenakan dataset *NASA public MDP* terdapat label kelas dalam setiap datanya, maka teknik diskritisasi pada penelitian ini adalah *supervised*. Salah satu metode diskritisasi *supervised* yang terbukti lebih unggul dari yang lain adalah *entropy-based discretization (EBD)* (Singh & Verma 2014) (Kohavi & Sahami 1996) (Singh & Verma 2009).

Algoritma *EBD* menggunakan nilai *entropy* informasi kelas dari kandidat pembagi untuk memilih batas diskrit (Dougherty et al. 1995). Secara umum *entropy* adalah ukuran dari sebuah ketidakpastian dengan variabel yang acak (Akalya Devi et al. 2012). Diasumsikan terdapat satu set *instance S* yang telah diurutkan secara *ascending* (dari nilai yang paling rendah ke nilai yang paling tinggi). Satu set *instance S* tersebut kemudian dipartisi menjadi subset *S1* dan *S2*. Adapun *entropy* kelas subset *S* didefinisikan sebagai berikut (Fayyad & Irani 1993):

$$Ent(S) = -\sum_{i=1}^{Z} p(C_i, S)\, log_2(p(C_i, S)) \tag{2.1}$$

Dimana $p(C_i,S)$ adalah proporsi *instance* dengan kelas C_i, sedangkan Z adalah jumlah kelas. *Entropy* kelas dihitung berdasarkan bobot rata-rata *entropy* individual, yaitu S_1 dan S_2 (hasil partisi *S*). *Entropy* informasi kelas dari partisi yang diinduksi oleh titik potong *T*, untuk sebuah fitur *F*, dihitung sebagai berikut:

$$E(F, T; S) = \frac{|S_1|}{S} Ent(S_1) + \frac{|S_2|}{S} Ent(S_2) \tag{2.2}$$

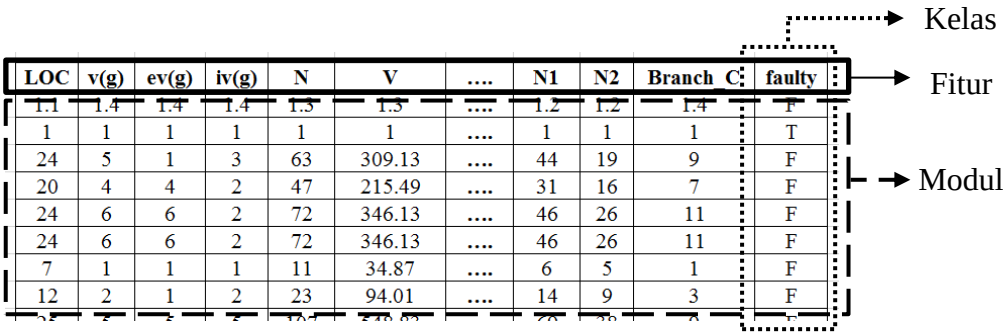
Nilai *T* adalah titik potong yang diambil dari titik tengah nilai fitur yang sudah diurutkan. Adapun titik potong *E(F,T;S)* yang paling minimum di antara semua kandidat titik potong, diambil sebagai titik potong terbaik T_F dan sebagai penentu diskritisasi biner dari sebuah fitur. Adapun titik potong minimum tersebut dipilih berdasarkan *Gain* yang terbesar. Semakin semakin kecil nilai titik potong *E(F,T;S)*, maka semakin besar pula nilai *Gain* yang didapat. Adapun untuk memperoleh *Gain* dapat dilihat pada rumus di bawah ini:

$$Gain(F,T;S) = Ent(S) - E(F,T;S) \quad (2.3)$$

Semua kemungkinan *split* dievaluasi secara rekursif, kemudian dipilih *split* terbaik yaitu menghasilkan nilai *gain* tertinggi *(maximum Gain)*. Sebuah titik *split* ditentukan oleh diskritisasi yang paling dapat meminimalkan fungsi *E*. Setelah proses *EBD* selesai, data hasil diskritisasi akan digunakan oleh 5 metode seleksi fitur untuk dilakukan pemeringkatan fitur berdasarkan *score* dari masing-masing metode. Nantinya setelah seleksi fitur dilakukan, akan ada 5 versi peringkat fitur dari 5 metode seleksi fitur yang berbeda-beda.

2.5. Seleksi Fitur

Fitur atau atribut atau variabel atau matrik merujuk kepada aspek sebuah data. Fitur adalah properti individu terukur dari proses yang diamati (Chandrashekar & Sahin 2014). Umumnya sebelum proses pengumpulan data dilakukan, fitur ditentukan dan dipilih terlebih dahulu. Setiap fitur dapat terdiri dari fitur diskrit, kontinyu, atau nominal. Sebagai contoh, fitur pada dataset kesalahan perangkat lunak dapat dilihat pada Gambar 2.2.



LOC	v(g)	ev(g)	iv(g)	N	V	....	N1	N2	Branch_C	faulty
1.1	1.4	1.4	1.4	1.3	1.3	....	1.2	1.2	1.4	F
1	1	1	1	1	1	....	1	1	1	T
24	5	1	3	63	309.13	....	44	19	9	F
20	4	4	2	47	215.49	....	31	16	7	F
24	6	6	2	72	346.13	....	46	26	11	F
24	6	6	2	72	346.13	....	46	26	11	F
7	1	1	1	11	34.87	....	6	5	1	F
12	2	1	2	23	94.01	....	14	9	3	F

Gambar 2.2 Contoh Fitur Pada Dataset Kesalahan Perangkat Lunak

Berdasarkan Gambar 2.2, fitur-fitur dataset kesalahan perangkat lunak umumnya terdiri dari fitur kontinyu (numerik). Setiap baris pada dataset mewakili modul perangkat lunak. Sedangkan kelas adalah keterangan pada setiap modul di dataset, apakah terdapat kesalahan pada modul atau tidak (diskrit). Namun, berdasarkan penelitian sebelumnya, penggunaan semua fitur pada dataset sebagai input model prediksi belum tentu dapat meningkatkan nilai akurasi (Singh et al. 2016) (Menzies et al. 2007). Salah satu faktor penyebabnya adalah kualitas data dari setiap fitur (Abraham et al. 2009). Fitur-fitur yang dapat menurunkan nilai

akurasi model prediksi adalah fitur yang tidak relevan dan fitur yang redundan. Oleh karena itu, hanya fitur-fitur relevan dan tidak redundan saja yang akan dijadikan input model prediksi (Abraham et al. 2009).

Berdasarkan karakteristiknya, fitur terbagi menjadi tiga, yaitu fitur yang relevan, fitur yang tidak relevan, dan fitur yang redundan (Ladha & Deepa 2011). Fitur yang relevan adalah fitur yang memiliki pengaruh pada *output* dan peran fitur tersebut tidak dapat diasumsikan oleh fitur lainnya. Sedangkan fitur yang tidak relevan didefinisikan sebagai fitur yang tidak memiliki pengaruh pada *output*. Misalnya fitur *id* pada dataset, dikarenakan nilai *id* bukan aspek terukur pada sebuah data, maka fitur *id* tidak memiliki pengaruh pada *output* model prediksi. Adapun fitur yang redundan adalah fitur dapat mengambil peran dari fitur yang lain. Contoh fitur yang redundan adalah sebagai berikut:
Diasumsikan bahwa fitur *N1* adalah jumlah *operator*, fitur *N2* adalah jumlah *operand*, sedangkan fitur *N* adalah total dari nilai fitur *N1* dan *N2*. Jika fitur *N* dihilangkan, maka peran fitur *N* masih dapat diwakilkan oleh fitur *N1* dan *N2*. Ini artinya fitur *N* adalah fitur yang redundan dan tidak perlu untuk dipilih sebagai *input* model prediksi.

Proses pemilihan fitur yang relevan pada sebuah dataset disebut seleksi fitur. Seleksi fitur adalah pra-proses yang penting untuk diterapkan pada dataset sebelum diterapkan algoritma klasifikasi, karena metode seleksi fitur dapat menghilangkan fitur yang redundan maupun yang tidak relevan. Istilah seleksi fitur mengacu pada algoritma yang dapat menghasilkan sebuah subset dari satu set *input* fitur. Pengurangan dimensi data juga dapat mengurangi ruang hipotesis dan memungkinkan algoritma untuk beroperasi dengan lebih cepat dan efektif. Pada penelitian ini beberapa metode seleksi fitur akan diintegrasikan dengan metode *CBC* untuk memprediksi kesalahan perangkat lunak.

Algoritma seleksi fitur yang diterapkan pada tahap pra-proses secara umum terbagi menjadi dua kategori yaitu *filter* dan *wrapper* (Karegowda et al. 2010) (Yang et al. 2011) (Abaei & Selamat 2014). Metode *filter* menggunakan teknik peringkat *(ranking)* pada fitur sebagai kriteria dasar untuk urutan seleksi fitur (Chandrashekar & Sahin 2014). Metode peringkat digunakan karena kesederhanaan prosesnya dalam melakukan seleksi fitur. Tidak hanya itu, beberapa penelitian telah

membuktikan metode seleksi fitur dengan peringkat dapat meningkatkan akurasi prediksi kesalahan perangkat lunak (Singh et al. 2016). Umumnya metode ini memanfaatkan nilai *threshold* untuk menentukan berapa banyak jumlah minimal fitur yang akan digunakan sebagai input pada model prediksi. Jumlah minimal tersebut diurutkan berdasarkan peringkat fitur. Fitur dengan nilai tertinggi akan menjadi kandidat input pemodelan prediksi. Beberapa metode seleksi fitur menggunakan filter yaitu *information gain (IG), gain ratio (GR), one-r (OR), relief-f (RFF),* dan *symmetric uncertainty (SU).*

Metode *wrapper* menggunakan teknik komputasi kompleks yang didasarkan pada algoritma klasifikasi kompleks. Metode ini akan mencoba beberapa atau bahkan semua kemungkinan kombinasi fitur untuk mengevaluasi hasil akurasi prediksi (Akalya Devi et al. 2012). Proses seleksi fitur akan berkembang secara eksponensial pada jumlah fitur yang lebih banyak. Model komputasi *wrapper* menjadi sangat intensif terhadap dataset yang memiliki dimensi yang besar.

Pada penelitian ini, pendekatan seleksi fitur dilakukan dengan metode *filter*. Berdasarkan (Gao et al. 2011), penggunaan metode *wrapper* akan menjadi sangat kompleks jika diterapkan pada dataset kesalahan perangkat lunak, mengingat dimensi dataset yang besar. Selain itu, dependensi *wrapper* terhadap model klasifikasi tertentu membuat sulitnya menemukan *wrapper* terbaik dan yang paling cocok terhadap kasus ini, karena terdapat beragam model klasifikasi yang tersedia untuk dipilih. Adapun perbandingan fitur pada setiap 7 dataset *NASA public MDP* dapat dilihat di Tabel 2.3.

Tabel 2.3 Perbandingan Fitur pada 7 Dataset *NASA Public MDP*

No	Kategori	Fitur	Representasi	Dataset NASA public MDP						
				CM1	KC3	MW1	PC1	PC2	PC3	PC4
1	line of code	LOC total	LOC	x	x	x	x	x	x	x
2		LOC blank	BLOC	x	x	x	x		x	x
3		LOC executable	SLOC	x	x	x	x	x	x	x
4		LOC comments	CLOC	x	x	x	x	x	x	x
5		LOC code and comment	C&SLOC	x	x	x	x	x	x	x
6		Number of lines	nl		x	x		x	x	x
7		Percent comment	% comment		x	x		x	x	x
8	halstead	Number of operators	N1	x	x	x	x	x	x	x
9		Number of operands	N2	x	x	x	x	x	x	x
10		Number operators + operands	N	x			x			
11		Number of unique operators	n1	x	x	x	x	x	x	x
12		Number of unique operands	n2	x	x	x	x	x	x	x
13		Length	L	x	x	x	x	x	x	x
14		Difficulty	D	x	x	x	x	x	x	x
15		Level	1/D		x	x		x	x	x
16		Volume	V	x	x	x	x	x	x	x
17		Programming effort	E	x	x	x	x	x	x	x
18		Programming time	T	x	x	x	x	x	x	x
19		Error estimate	B	x	x	x	x	x	x	x
20		Content (intelligence) vocabulary	I	x	x	x	x	x	x	x
21	mcCabe	Cyclomatic complexity	v(G)	x	x	x	x	x	x	x
22		Cyclomatic density	vd(G)		x	x		x	x	x
23		Decision density	dd(G)		x	x		x	x	x
24		Design complexity	iv(G)	x	x	x	x	x	x	x
25		Design density	id(G)		x	x		x	x	x
26		Essential complexity	ev(G)	x	x	x	x	x	x	x
27		Essential density	ed(G)		x	x		x	x	x
28		Global data complexity	gdv		x					
29		Global data density	gd(G)		x					
30		Norm cyclomatic compl	Normv(G)		x	x		x	x	x
31		Maintenance severity	Mainsev		x	x		x	x	x
32	miscellaneous	Branch count	Branch_C	x	x	x	x	x	x	x
33		Call pairs	Call_C		x	x		x	x	x
34		Condition count	Cond_C		x	x		x	x	x
35		Decission count	Dec_C		x	x		x	x	x
36		Edge count	Edge_C		x	x		x	x	x
37		Node count	Node_C		x	x		x	x	x
38		Parameter count	Parameter_C		x	x		x	x	x
39		Multiple condition count	Mul_Cond_C		x	x		x	x	x
40		Modified condition count	Mod_Cond_C		x	x		x	x	x
41	classification	Faulty module	faulty (true/false)	x	x	x	x	x	x	x
			Jumlah fitur	22	40	38	22	37	38	38

Pada Tabel 2.3 terdapat lima kategori fitur utama yaitu *line of code*, *hastead*, *mcCabe*, *miscellaneous* dan *classification*. Perbedaan kategori *halstead* dan *mcCabe* adalah asumsi prediksi kesalahan perangkat lunak. Matrik *halstead* beranggapan bahwa kode yang sulit dibaca lebih mungkin terjadi kesalahan. Sedangkan matrik *mcCabe* beranggapan bahwa kode dengan jalur yang rumit lebih rawan terhadap kesalahan.

Dikarenakan metode seleksi fitur pada penelitian ini adalah *filter*, maka fitur-fitur dari setiap dataset akan dilakukan pemeringkatan. Setiap fitur akan dihitung *score*-nya berdasarkan metode seleksi fitur *IG*, *GR*, *OR*, *RFF*, dan *SU*. Setiap metode seleksi fitur memiliki penilaian peringkat *(ranking)* yang berbeda. Fitur dipilih berdasarkan *Top X ranking* (*X* peringkat tertinggi) pada setiap metode seleksi fitur. Untuk melakukan validasi terhadap fitur yang diseleksi, maka hasil seleksi fitur digunakan sebagai *input* model prediksi. Adapun ilustrasi *scoring* dan pemilihan *Top X ranking* dapat dilihat pada Tabel 2.4.

Tabel 2.4 Ilustrasi *Scoring* dan Pemilihan Dua *Ranking* Tertinggi

Ranking	Fitur	*Score*	
1	Number of operator	0,38	→ Fitur yang dipilih
2	Number of operand	0,24	
3	Nummber operators + operands	0,12	

Berdasarkan Tabel 2.4, diasumsikan terdapat tiga fitur pada dataset. Setiap fitur dilakukan perhitungan *score* menggunakan metode seleksi fitur. Setelah itu fitur diurutkan berdasarkan *score* tertinggi. Proses pemilihan dilakukan dengan mengambil dua fitur tertinggi. Semakin tinggi nilai *score*, berarti semakin tinggi pula relevansi fitur terhadap *output* prediksi. Sub-bab selanjutnya, dijelaskan metode-metode perhitungan *score* untuk melakukan seleksi fitur pada dataset *NASA public MDP* yaitu *IG*, *GR*, *OR*, *RFF*, dan *SU*.

2.5.1. Information Gain (IG)

IG adalah sejumlah informasi yang diperoleh dengan mengetahui nilai fitur. Artinya, jika *entropy* itu adalah nilai ketidakpastian dari sebuah fitur, maka *IG* adalah pengurangan nilai yang diharapkan dalam *entropy*. Semakin kecil nilai ketidakpastian *(entropy)*, maka semakin besar pula informasi yang akan didapat *(gained)*. Dikarenakan *IG* mampu menilai pentingnya fitur dengan mengukur nilai informasi yang didapat (apakah berhubungan dengan label kelas atau tidak), maka *IG* dapat mengubah nilai ketidakpastian sebuah informasi *(entropy)* menjadi ukuran nilai informasi yang akan didapat sebelum akhirnya informasi tersebut diambil dari beberapa fitur. Karakteristik *IG* yaitu memerlukan data diskrit dalam melakukan pemeringkatan fitur. Adapun rumus seleksi fitur menggunakan *IG* adalah sebagai berikut:

$$IG(Class, a) = E(Class) - E(Class|a) \quad (2.4)$$

dimana *E* adalah nilai *entropy*. Diasumsikan *A* sebagai himpunan semua fitur, *Class* sebagai fitur dependen dari semua *training*, nilai (*a*,*y*) dengan $y \in Class$ mendefinisikan nilai dari contoh spesifik untuk fitur $a \in A$, *v* merepresentasikan satu set nilai-nilai dari fitur *a*, yaitu $v = \{value(a,y) \mid a \in A \cap y \in Class\}$. Rumus *IG* pada setiap fitur $a \in A$ didefinisikan sebagai berikut:

$$\begin{aligned} IG(Class, a) = {} & E(Class) \\ & - \sum_{v \in V} \frac{|\{y \in Class | value(a, y) = v\}|}{|Class|} \\ & \times E(\{y \in Class | value(a, y) = v\}) \end{aligned} \quad (2.5)$$

Pada umumnya *IG* dapat memberikan ukuran yang baik dalam menentukan relevansi atribut, tetapi terdapat keterbatasan yaitu bias yang mendukung fitur dengan banyak pilihan data. Misalnya, terdapat beberapa data yang menggambarkan modul proyek perangkat lunak. Salah satu fitur yang mungkin pada proyek perangkat lunak adalah *id*. fitur ini dapat memiliki nilai *IG* yang tinggi, tetapi pada kenyataannya *id* tidak digunakan dalam proses prediksi kesalahan perangkat lunak (Gao et al. 2011).

2.5.2. Gain Ratio (GR)

GR adalah hasil modifikasi *IG* untuk mengurangi bias fitur yang memiliki banyak pilihan data. Nilai *GR* akan tinggi jika persebaran data rata, dan akan bernilai rendah ketika semua data masuk ke dalam masing-masing satu pilihan data. *GR* memodifikasi *IG* dengan memperhatikan jumlah hasil yang diperoleh dari kondisi pengujian atribut (Gao et al. 2011). Karakteristik *GR* tidak jauh berbeda dengan *IG* yaitu sama-sama memerlukan data diskrit dalam melakukan pemeringkatan fitur. Adapun rumus *GR* adalah sebagai berikut:

$$GR(Class, a) = \frac{IG(Class, a)}{E(a)} \quad (2.6)$$

dengan perhitungan *E(a)* sebagai berikut:

$$E(a) = -\sum_{v \in V} \frac{|\{y \in Class | value(a,y) = v\}|}{|Class|} \times log_2\left(\frac{|\{y \in Class | value(a,y) = v\}|}{|Class|}\right) \quad (2.7)$$

Semua notasi rumus pada *GR* sama seperti rumus pada *IG*. *E(a)* disebut juga sebagai *split info* (Quinlan 1986).

2.5.3. One-R (OR)

OR adalah algoritma klasifikasi sederhana yang menghasilkan sebuah aturan untuk setiap prediktor dalam data, kemudian memilih aturan dengan total kesalahan terkecil sebagai *"one rule"*. Untuk membuat aturan *OR*, perlu dilakukan perhitungan frekuensi untuk masing-masing kelas terhadap data.

Seleksi fitur *OR* dapat membangun aturan berdasarkan fitur tunggal untuk setiap fitur dalam kumpulan data. Dengan memecah data ke dalam *training* dan *testing*, *OR* memungkinkan untuk menghitung skor akurasi klasifikasi untuk setiap fitur. Penelitian sebelumnya, telah memilih fitur dengan skor tertinggi dan menunjukkan bahwa untuk sebagian besar dataset *UCI* memiliki keterkaitan dengan fitur tunggal *(OR)*. Berdasarkan teori tersebut, *OR* dapat digunakan sebagai filter untuk mengurangi fitur pada dataset. Karakteristik *OR* yaitu memerlukan data diskrit dalam melakukan pemeringkatan fitur. Adapun algoritma *OR* adalah sebagai berikut:

For each fitur *f*,

For each nilai *v* dari domain *f*

Pilih set *instance* dengan fitur *f* mempunyai nilai *v*

Diasumsikan *c* adalah kelas yang memiliki frekuensi paling tinggi

Terapkan "*if* fitur *f* memiliki nilai *v then* kelas *is c*" pada fitur *f*

Output aturan dengan akurasi klasifikasi tertinggi.

2.5.4. Relief-F (RFF)

RFF adalah perbaikan metode dari *relief*, dimana *relief* sendiri adalah metode estimasi pembobotan sebuah fitur. Semakin besar bobot sebuah fitur, maka dianggap semakin relevan fitur tersebut dengan *output*. Namun, *relief* sudah lama tidak digunakan lagi karena ketidakstabilan akurasi yang dihasilkan, dikarenakan ketidakmampuannya untuk mengambil dan mengevaluasi sampel berulang kali

dengan bobot fitur yang sama (R.P.L.DURGABAI 2014). Sedangkan *RFF* dapat mengevaluasi nilai fitur dengan berulang kali mengambil sampel *instance* dan mempertimbangkan nilai fitur yang diberikan untuk *instance* terdekat dari kelas yang sama dan yang berbeda. Evaluasi atribut ini memberikan bobot untuk masing-masing fitur berdasarkan kemampuan fitur untuk membedakan antar kelas, dan kemudian memilih fitur-fitur yang nilainya melebihi *threshold* yang ditetapkan sebelumnya sebagai fitur yang relevan (Novakovic 2010). Pemilihan atribut dilakukan dengan menghitung perbedaan bobot untuk tiap fitur yang terpilih secara acak (random sampling) dengan fitur yang terpilih sebagai *near hit* (tetangga terdekat fitur terpilih pada kelas yang sama) dan *near miss* (tetangga terdekat fitur terpilih pada kelas yang berbeda) (Irawan et al. 2011).

Perhitungan *threshold* dilakukan berdasarkan pada probabilitas *nearest neighbors* dari dua kelas berbeda dengan nilai yang berbeda untuk sebuah fitur dan probabilitas dua *nearest neighbors* dari kelas yang sama dan fitur dengan nilai yang sama. Semakin tinggi perbedaan nilai antara dua probabilitas, semakin signifikan pula fitur tersebut. Karakteristik *RFF* yaitu tidak hanya dapat menggunakan data diskrit saja dalam melakukan pemeringkatan fitur, tetapi juga *RFF* dapat menangani data numerik. Adapun algoritma *RFF* adalah sebagai berikut (Yang et al. 2011):

Input: sebuah *training* set *D*, jumlah iterasi *m*, jumlah *nearest neighbors* k, jumlah fitur *n, predefine* bobot fitur *threshold* δ.

Output: subset fitur *S* yang didasari oleh fitur yang bobotnya lebih besar dari *threshold* δ.

Langkah 1.

Diasumsikan $S=\emptyset$, set semua bobot fitur $W(F_t)=0$, t=1,2,…,*n*.

Langkah 2.

(1) Pilih *sample R* dari *D* secara acak.

(2) Cari *k nearest neighbors* H_i (*i*=1,2,…,*k*) dari kelas yang sama dan *k nearest neighbors* M_i (C)(*i*=1,2,…,k) dari setiap kelas *C* yang berbeda.

(3) For *t* = 1 to *n* do

$$W(F_t) = W(F_t) - \sum_{i=1}^{k} \frac{diff(F_t, R, H_i)}{(mk)} + \sum_{C \notin Class(R)} \frac{\left(\frac{P(C)}{1-P(Class(R))} \sum_{i=1}^{k} diff\left(F_t, R, M_i(C)\right)\right)}{(mk)} \quad (2.8)$$

Langkah 3. For *t* = 1 to *n* do

If $W(F_t) > \delta$, then add fitur F_t to *S*.

Pada (1), *P(C)* adalah distribusi probabilitas class *C*, *Class(R)* adalah kelas yang masuk ke dalam kategori *R*, sedangkan M_i(C) menunjukkan *i near miss* dari *R* dalam kelas *C*, *diff*(F_t, R_1, R_2) menunjukkan perbedaan R_1 dan R_2 pada F_t.

If F_t adalah data diskrit:

$$diff(F_t, R_1, R_2) = \begin{cases} 0; R_1[F_t] = R_2[F_t] \\ 1; R_1[F_t] \neq R_2[F_t] \end{cases}$$

If F_t adalah data kontinyu:

$$diff(F_t, R_1, R_2) = \frac{|R_1[F_t] - R_2[F_t]|}{\max(F_t) - \min(F_t)}$$

dimana R_1 dan R_2 adalah dua sampel, $R_1[F_t]$ dan $R_2[F_t]$ adalah nilai fitur dari masing-masing R_1 dan R_2 pada F_t.

2.5.5. Symmetric Uncertainty (SU)

SU adalah pengukuran validitas klasifikasi matrik berdasarkan *entropy*. Peningkatan akurasi dilakukan tanpa mempengaruhi fungsi relatif dari kelas-kelas pada dataset (Antony et al. 2016). *SU* juga mengkompensasi bias *IG* terhadap fitur dengan nilai lebih tersendiri dan menormalkan nilai-nilai ke kisaran 0 hingga 1. Nilai 1 menunjukkan bahwa pengetahuan tentang fitur relevan terhadap kelas, sedangkan nilai 0 sebaliknya, fitur tersebut artinya independen. *SU* menggunakan perhitungan fitur diskrit dari *IG* (Yang et al. 2011). Karakteristik *SU* yaitu memerlukan data diskrit dalam melakukan pemeringkatan fitur. Adapun pengukuran *SU* dapat dihitung dengan rumus berikut:

$$SU(Class, a) = \frac{2 \times IG(Class, a)}{E(Class) + E(a)} \quad (2.9)$$

Dikarenakan kemiripan rumus *SU* terhadap *IG* dan *GR*, maka hasil dapat menunjukkan nilai yang tidak jauh berbeda (Singh et al. 2016). Namun pada penelitian yang lain, hasil perhitungan *SU* berbeda dengan yang *IG* dan *GR* (Gao et al. 2011).

2.6. *Cluster-based Classification (CBC)*

Clustering adalah salah satu metode untuk menemukan kelompok yang memiliki paling banyak kesamaan dari data yang diberikan, yang berarti data dapat termasuk ke dalam satu kelompok yang paling mirip dan data yang termasuk ke dalam kelompok berbeda adalah data yang paling berbeda. *Cluster-based classification (CBC)* adalah metode klasifikasi yang memanfaatkan jarak pada setiap data dari titik *cluster*. Data yang memiliki tingkat kemiripan yang tinggi dengan titik *cluster* maka dianggap sebagai anggota *cluster* tersebut. Dalam kasus prediksi kesalahan perangkat lunak, *cluster* dibagi menjadi dua yaitu kelompok data yang rentan terhadap salah dan yang tidak. Berbagai algoritma *clustering* telah diusulkan oleh peneliti sebelumnya. Berdasarkan penelitian (Singh & Verma 2014), *k-means* dipilih sebagai metode *clustering* yang digunakan untuk klasifikasi.

Algoritma *k-means clustering* dimulai dengan sekumpulan data *training* dan sejumlah titik *cluster K*. Sampel *training* pada dataset digunakan untuk mengelompokkan data berdasarkan pengukuran kedekatan jarak terhadap titik *cluster*. Ada beberapa cara untuk mengukur jarak antar suatu obyek dengan obyek lainnya, antara lain *euclidean distance*, *manhattan distance*, dan *hamming distance*.

Euclidean distance sangat sering digunakan untuk menghitung jarak antara dua obyek. *Euclidean distance* menghitung akar pangkat dua dari perbedaan koordinat sepasang objek. Beberapa peneliti sebelumnya juga telah menggunakan *eulidean distance* untuk mencari jarak antar obyek (Singh & Verma 2014) (Singh & Vyas 2014). Sedangkan *Manhattan distance* merepresentasikan jarak antara dua obyek secara absolut (Murti et al. 2005). Baik *euclidean* maupun *manhattan*, keduanya digunakan untuk menghitung jarak data numerik.

Dikarenakan hasil pra-proses diskritisasi adalah data biner, maka perhitungan jarak yang akan digunakan pada penelitian ini adalah *hamming distance*. Berdasarkan penelitian sebelumnya, *hamming distance* digunakan untuk menghitung jumlah perbedaan dari dua deret bilangan biner yang mempunyai panjang yang sama sesuai dengan posisi dari setiap digit biner (Murti et al. 2005). Rumus *hamming distance* adalah sebagai berikut:

$$d_{ij} = q + r \tag{2.10}$$

Dimana *q* adalah jumlah variabel dengan nilai 1 pada objek ke-*i* tetapi bernilai 0 pada obyek ke-*j*. Sedangkan *r* adalah jumlah variabel dengan nilai 0 pada obyek ke-*i* tetapi bernilai 1 pada obyek ke-*j*.

Misalnya, titik A (1,1,1,1) dan B (0,1,0,0), maka *hamming distance* antara A dan B dapat dicari dengan cara:

q = 3 (bernilai 1 di A, tetapi bernilai 0 di B)

r = 0 (bernilai 1 di B, tetapi bernilai 0 di A)

$d_{(BA)}$ = 3 + 0 = 3

Hamming distance sangat berguna dalam pencarian jarak antar data biner hasil diskritisasi, khususnya dalam dataset *NASA public MDP*. Sedangkan algoritma *k-means clustering* adalah sebagai berikut:

Input	:	Diasumsikan terdapat satu set N data vektor x = (x_1 , ..., x_n)
K	:	Jumlah *cluster*
Output	:	Satu set K cluster terhadap data vektor
Inisialisasi	:	Pilih secara acak titik *K* sebagai titik pusat sebuah *cluster*
Perulangan	:	Hitung jarak *hamming* setiap data terhadap semua titik *cluster*
		Hitung ulang *centroid* dari tiap *cluster* berdasarkan nilai rata-rata
		Hingga tidak ada perubahan pada *centroid*

2.7. *Pd, Pf, dan Balance*

Terdapat tiga *output* pada penelitian ini, yaitu nilai probabilitas deteksi *(pd)*, nilai probabilitas *false alarm (pf)*, dan nilai *balance* dari *trade-off* antara kedua nilai probabilitas tersebut. *Pd* adalah nilai keberhasilan sistem dalam memprediksi kesalahan perangkat lunak. Sedangkan *pf* adalah nilai misklasifikasi sistem dalam

menentukan modul yang secara aktual tidak terdapat kesalahan, namun diklasifikasikan sebagai modul yang salah. Idealnya *output* sistem prediksi kesalahan perngkat lunak dapat menghasilkan nilai *pd* sebesar 100% dan nilai *pf* sebesar 0%. Artinya sistem dapat memprediksi secara tepat kesalahan perangkat lunak tanpa melakukan misklasifikasi terhadap modul yang tidak terdapat kesalahan perangkat lunak.

Nilai *pd* yang tinggi tidak dapat dijadikan tolak ukur bahwa keandalan sistem prediksi kesalahan perangkat lunak itu baik, karena bisa jadi nilai *pf* yang dihasilkan sistem prediksi juga tinggi. Jika begitu, maka sistem prediksi cenderung untuk menganggap sebagian besar modul adalah rentan terhadap kesalahan, walaupun secara aktual, sebagian besar modul tersebut tidak terdapat kesalahan. Kondisi tersebut dapat dianggap efektif dalam memprediksi kesalahan perangkat lunak, karena sistem prediksi cenderung berhasil dalam memperdiksi kesalahan. Tetapi, sumber daya dan biaya yang dikeluarkan untuk pengujian menjadi tidak efisien, karena intensitas *false alarm* dari sistem prediksi juga tinggi.

Nilai *pd* dan nilai *pf* yang ideal sulit untuk dicapai, mengingat persebaran jumlah kelas *false* dan kelas *true* pada dataset yang tidak seimbang. Oleh karena itu diperlukan nilai *balance* sebagai *trade-off* antara nilai *pd* dan nilai *pf* yang dihasilkan dari sistem prediksi kesalahan perangkat lunak. Semakin nilai *pd* mendekati 100% dan nilai *pf* mendekati 0%, maka semakin tinggi pula nilai *balance* yang dihasilkan sistem prediksi tersebut. Berikut adalah rumus yang digunakan untuk menghitung nilai *pd*, nilai *pf*, dan nilai *balance*:

$$pd = \frac{TP}{(TP + FN)} \tag{2.11}$$

$$pf = \frac{FP}{(FP + TN)} \tag{2.12}$$

$$balance = 1 - \frac{\sqrt{(0 - pf)^2 + (1 - pd)^2}}{\sqrt{2}} \tag{2.13}$$

Nilai *TP* (*True Positive)* diperoleh dari total prediksi modul yang terdapat kesalahan *(true)* dengan aktual modul juga terdapat kesalahan *(true)*. Nilai *FN (False Negative)* diperoleh dari total prediksi modul yang tidak terdapat kesalahan *(false)* dengan aktual modul terdapat kesalahan *(true)*.

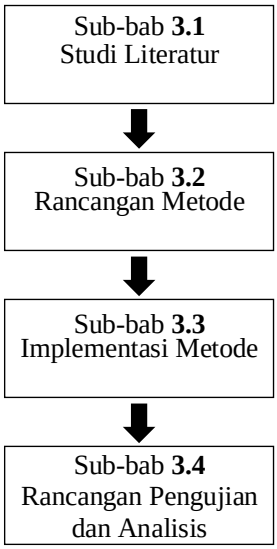
Sedangkan nilai *FP (False Positive)* adalah kebalikan dari *TP*, yaitu total prediksi modul yang tidak terdapat kesalahan *(false)* dengan aktual modul juga tidak terdapat kesalahan *(false)*. Nilai *TN (True Negative)* diperoleh dari total prediksi modul yang tidak terdapat kesalahan *(false)* dengan aktual modul tersebut terdapat kesalahan *(true)*. Adapun perolehan nilai *TP*, *FP*, *TN*, dan *FN* dapat dilihat di matrik *confusion* pada Tabel 2.5.

Tabel 2.5 Matrik *Confusion*

	AKTUAL	
PREDIKSI	*Faulty*	*Non-faulty*
Faulty	TP	FP
Non-faulty	FN	TN

BAB 3
METODOLOGI PENELITIAN

Untuk mencapai tujuan penelitian ini, ada beberapa langkah yang dilakukan, yaitu studi literatur, rancangan metode, rencana implementasi metode, serta rancangan pengujian dan analisis. Gambar 3.1 menunjukkan langkah-langkah yang dilakukan pada penelitian ini.



Gambar 3.1 Tahapan Penelitian

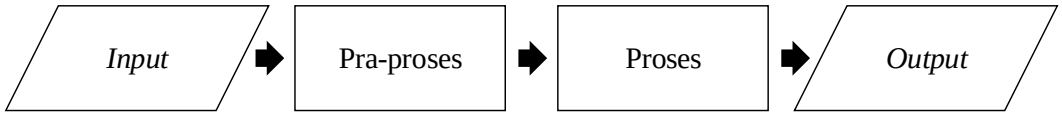
3.1. Studi Literatur

Pada tahap ini akan dipelajari tentang berbagai informasi dan sumber pustaka yang sesuai dengan konteks penelitian. Pada penelitian ini, literatur yang dikaji secara garis besar meliputi konsep-konsep dasar yang berkaitan dengan prediksi kesalahan perangkat lunak, metode diskritisasi menggunakan *EBD*, metode klasifikasi menggunakan *CBC*, serta lima metode seleksi fitur *IG*, *GR*, *OR*, *RFF*, dan *SU* untuk menghilangkan data yang berlebihan dan tidak relevan agar dapat meningkatkan akurasi prediksi pada kesalahan perangkat lunak.

3.2. Rancangan Metode

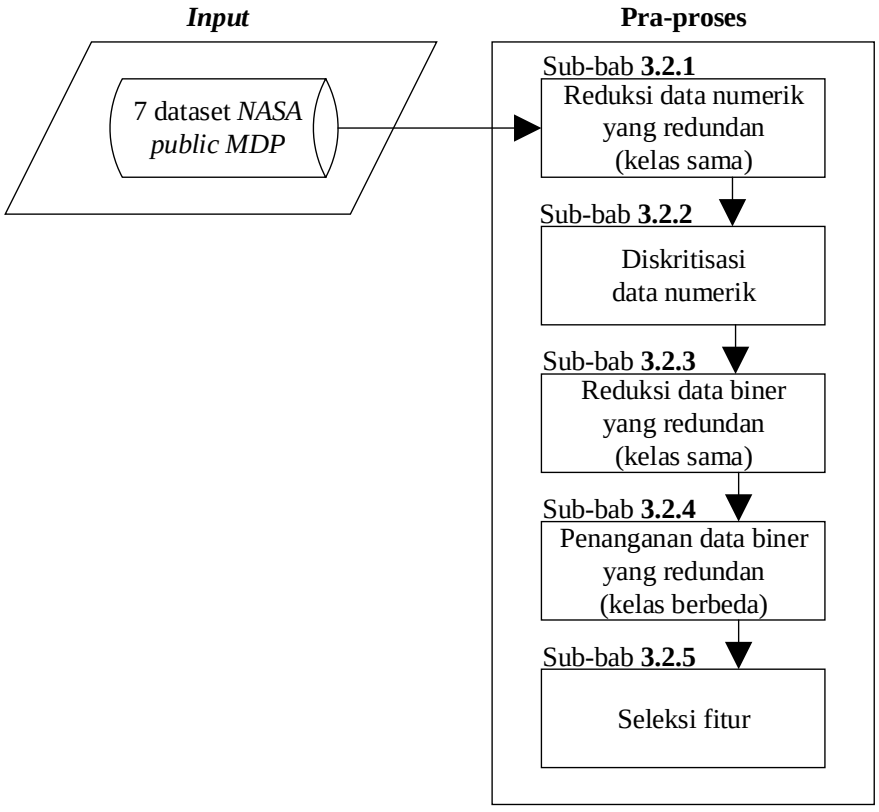
Rancangan metode terdiri dari tahap *input*, pra-proses, proses, dan *output*. *Input* pada penelitian ini adalah tujuh dataset *NASA public MDP* yaitu CM1, KC3,

MW1, PC1, PC2, PC3, PC4. *Input* tersebut dipilih berdasarkan penelitian sebelumnya (Singh & Verma 2014). Secara umum, rancangan metode pada penelitian ini disajikan pada Gambar 3.2.



Gambar 3.2 Rancangan Metode

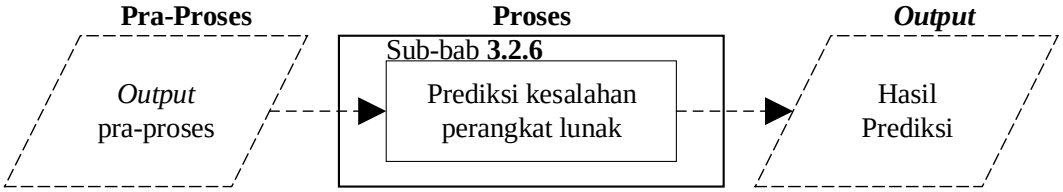
Pra-proses terdiri dari enam sub-proses yaitu reduksi data numerik yang redundan, diskritisasi data numerik, reduksi data biner yang redundan, penanganan data redundan dengan kelas yang berbeda, dan seleksi fitur. Dengan melakukan pra-proses, diharapkan akurasi prediksi kesalahan perangkat lunak pada dataset *NASA public MDP* dapat meningkat. Kegiatan pra-proses pada penelitian ini dapat dilihat pada Gambar 3.3.



Gambar 3.3 Urutan Kegiatan Pra-Proses

Hasil pra-proses tersebut nantinya digunakan sebagai *input* pada model prediksi kesalahan perangkat lunak. Pada penelitian sebelumnya (Singh & Verma 2014), pra-proses belum mencakup seleksi fitur. Sedangkan pada penelitian ini,

lima metode seleksi fitur diusulkan untuk diintegrasikan dengan model prediksi kesalahan perangkat lunak. Adapun tahapan proses pada penelitian ini dapat dilihat pada Gambar 3.4. *Output* dari proses ini adalah hasil prediksi kesalahan perangkat lunak *true* atau *false*.



Gambar 3.4 Urutan Kegiatan Proses

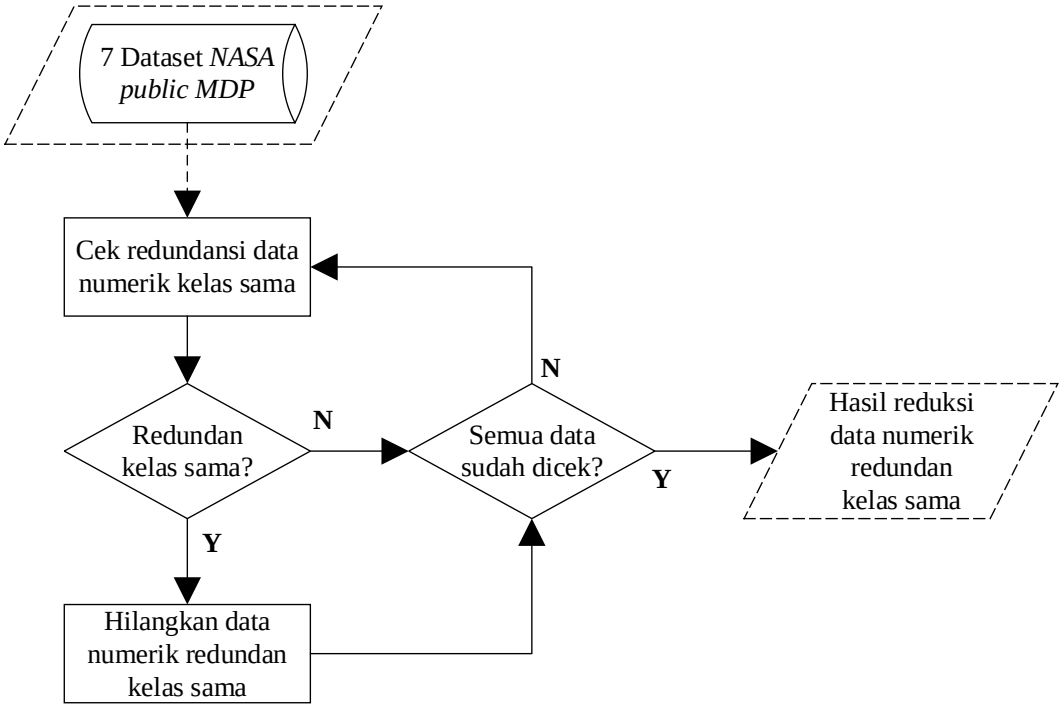
3.2.1. Reduksi Data Numerik yang Redundan dengan Kelas yang Sama

Menurut penelitian sebelumnya (Akalya Devi et al. 2012), data yang redundan perlu dihilangkan dulu karena akan berdampak pada penurunan nilai prediksi. Semua fitur pada 7 dataset *NASA public MDP* bertipe data numerik. Ketujuh dataset tersebut terdapat redundansi pada datanya. Redundansi yang dimaksudkan pada tahap ini adalah baris data numerik yang sama, termasuk pada label kelasnya. Di 7 dataset *NASA public MDP*, tidak ditemukan redundansi data numerik dengan kelas yang berbeda, jadi pada penelitian ini proses reduksi hanya dilakukan pada data numerik dengan kelas yang sama. Adapun contoh redundansi data numerik dengan kelas yang sama dapat dilihat pada Tabel 3.1.

Tabel 3.1 Contoh Redundansi Data Numerik dengan Kelas yang Sama

No	loc	iv(g)	e	b	brachCount	faulty
1	**24**	**3**	**2936,77**	**0,1**	**9**	**F**
2	20	2	3447,89	0,07	7	F
3	**24**	**3**	**2936,77**	**0,1**	**9**	**F**
4	12	1	2369,07	0,08	5	T

Pada Tabel 3.1 terdapat potongan 6 fitur dari dataset *NASA public MDP* yaitu *loc*, *iv(g)*, *e*, *b*, *branchCount*, dan *faulty*. Sedangkan *faulty* adalah kelas pada dataset *NASA public MDP*. Baris yang berwarna abu-abu adalah data yang redundan, yaitu baris ke 1 dan baris ke 3. Selanjutnya data yang redundan tersebut perlu dihilangkan. Adapun urutan proses reduksi data numerik yang redundan dengan kelas yang sama dapat dilihat di Gambar 3.5.

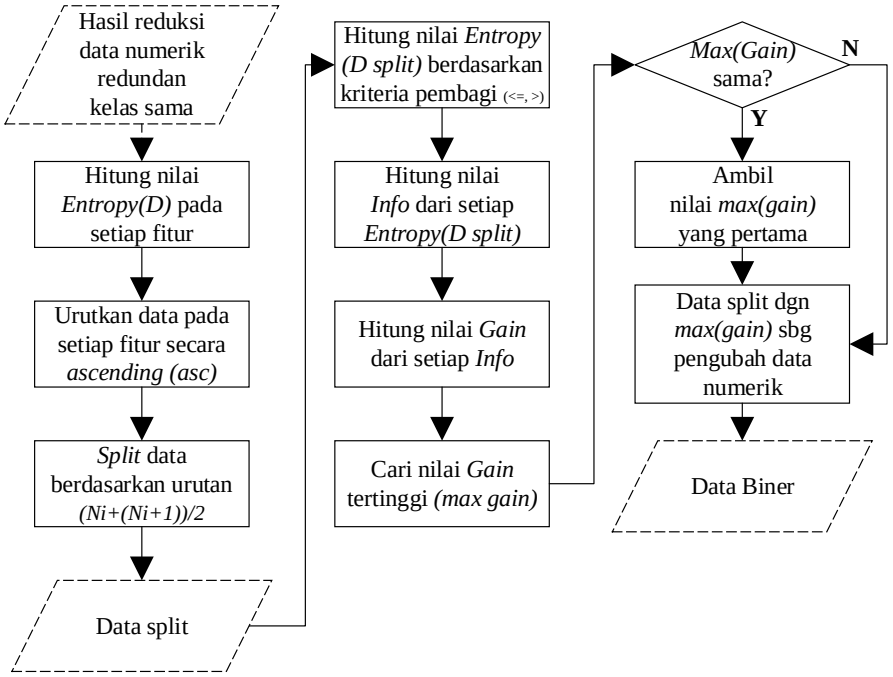


Gambar 3.5 Reduksi Data Numerik yang Redundan dengan Kelas yang Sama

Pada Gambar 3.5, setiap dataset akan dicek terlebih dahulu apakah terdapat data numerik redundan dengan kelas yang sama atau tidak. Jika terdapat redundansi data numerik dengan kelas yang sama, maka hanya salah satu data saja yang akan digunakan sebagai *input* di tahap selanjutnya.

3.2.2. Diskritisasi Data Numerik

Masing-masing dataset yang sudah melewati tahapan reduksi data numerik yang redundan dengan kelas yang sama, selanjutnya akan diubah ke data diskrit (biner). Perubahan data numerik ke data diskrit (biner) dilakukan untuk dapat meningkatkan akurasi prediksi kesalahan perangkat lunak dan meningkatkan efisiensi pada proses komputasi (Singh & Verma 2014) (Singh & Vyas 2014). Adapun langkah-langkah yang dilakukan pada tahap *EBD* disusun pada Gambar 3.6.



Gambar 3.6 Diskritisasi Data Numerik Menggunakan *EBD*

Berdasarkan Gambar 3.6, perhitungan nilai *entropy* dilakukan tiga kali pada setiap atribut, yaitu nilai *entropy* pada sebuah fitur, serta nilai *entropy* pada pembagi data dengan kriteria kurang dari sama dengan (<=) dan lebih dari (>). Data *split* yang memiliki nilai *gain* tertinggi, maka akan dijadikan kandidat utama untuk mengubah data numerik ke data disktrit. Adapun contoh *rule* diskritisasi per fitur adalah sebagai berikut:

Misalnya, angka 28 adalah hasil *split* pada data *training* di fitur *Y*. Jika angka 28 itu menghasilkan nilai *gain* tertinggi, maka data numerik yang kurang dari sama dengan (<=) 28 di fitur *Y*, akan diubah menjadi angka biner nol (0). Sebaliknya, angka yang lebih dari (>) 28 di fitur *Y* diubah menjadi angka biner satu (1). Untuk menjaga kemurnian informasi dari sebuah data, maka label pada kelas tidak akan diubah. Untuk lebih jelasnya, *rule* per fitur yang dijadikan acuan dalam pengubahan data numerik menjadi data biner di tahap *EBD* dapat dilihat pada Lampiran 2.1 hingga Lampiran 2.3. Adapun contoh hasil diskritisasi data numerik menjadi data biner dengan *EBD* dapat dilihat pada Tabel 3.2.

Tabel 3.2 Contoh Hasil Diskritisasi Data Numerik Menggunakan *EBD*

No	loc	iv(g)	e	b	branchCount	faulty
1	1	0	1	1	1	F
2	0	0	0	1	0	T
3	1	0	1	0	1	F

No	loc	iv(g)	e	b	branchCount	faulty
4	1	0	1	1	1	T
5	1	1	1	1	1	T
6	1	0	1	1	1	F

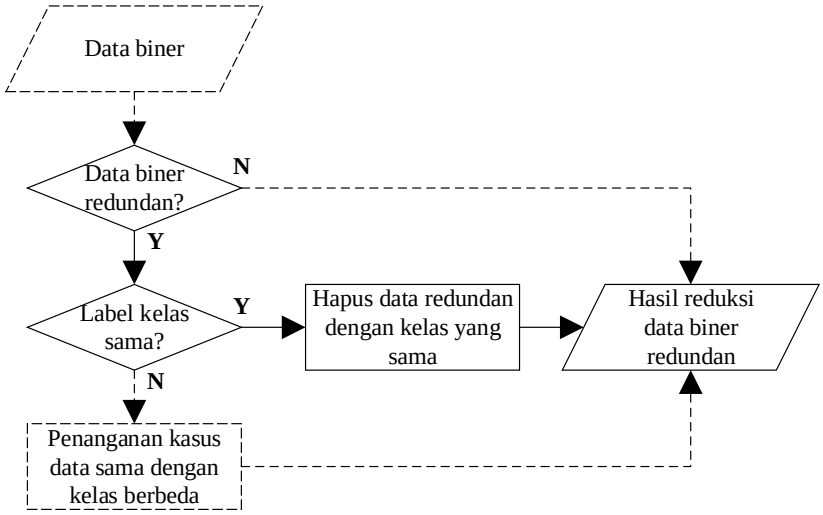
3.2.3. Reduksi Data Biner yang Redundan dengan Kelas yang Sama

Data redundan dapat mempengaruhi kinerja dan hasil prediksi kesalahan perangkat lunak (Akalya Devi et al. 2012). Pada tahap ini, data biner hasil diskritisasi yang redundan akan dihilangkan. Contoh redundansi data biner dengan kelas yang sama dapat dilihat pada Tabel 3.3.

Tabel 3.3 Contoh Data Biner dengan Kelas yang Sama

No	loc	iv(g)	e	b	branchCount	faulty
1	**1**	**0**	**1**	**1**	**1**	**F**
2	0	0	0	1	0	T
3	1	0	1	0	1	F
4	1	0	1	1	1	T
5	1	1	1	1	1	T
6	**1**	**0**	**1**	**1**	**1**	**F**

Pada Tabel 3.3 ditunjukkan bahwa data biner di baris ke 1 dan baris ke 6 adalah redundan. Data yang seperti itu perlu untuk direduksi. Redundansi data biner dengan kelas yang berbeda tidak dianggap sebagai redundan, karena informasi data berbeda (contoh: baris ke 1 dan baris ke 4). Kasus tersebut akan dijelaskan pada proses selanjutnya (Sub-bab 3.2.4). Adapun tahapan pada proses ini dapat dilihat pada Gambar 3.7.



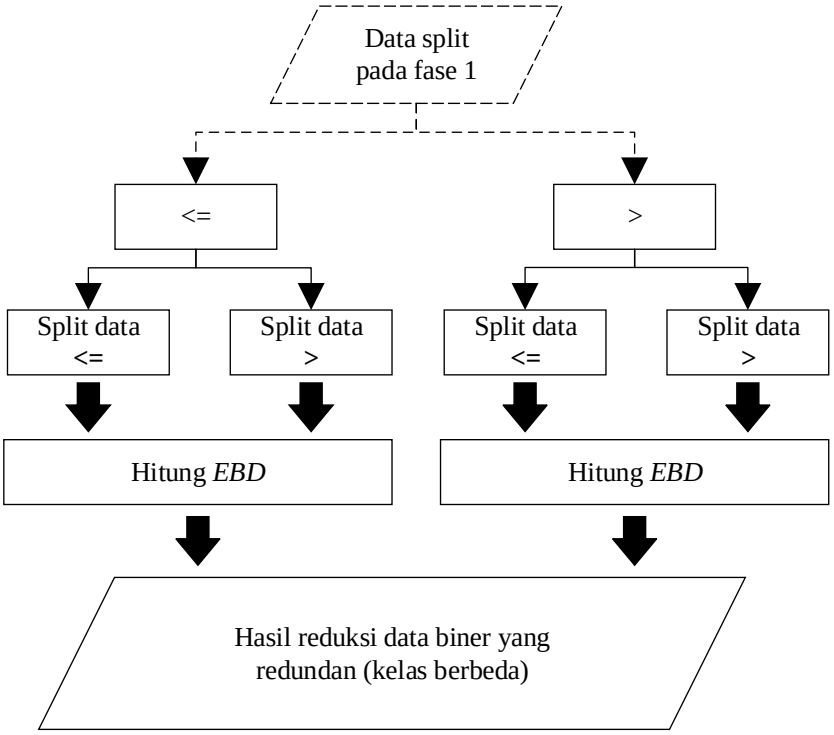
Gambar 3.7 Reduksi Data Biner yang Redundan dengan Kelas yang Sama

Proses pada Gambar 3.7 tidak jauh berbeda dengan proses di Gambar 3.5, yaitu jika terdapat kesamaan data di baris ke satu dengan baris data lainnya, maka

hanya akan digunakan salah satu data saja. Adapun *input* data yang digunakan pada Gambar 3.7 adalah data biner satu digit.

3.2.4. Penanganan Data Biner Redundan Dengan Kelas yang Berbeda

EBD mengubah data numerik menjadi angka biner dengan perhitungan *entropy*. Namun, dikarenakan tingginya interval data pada setiap dataset kesalahan perangkat lunak, jika dilakukan diskritisasi maka besar kemungkinan ditemukan duplikasi data diskrit dengan kelas yang berbeda. Jika hal itu terjadi, maka model prediksi dapat menjadi tidak konsisten. Oleh karena itu, untuk mengatasi masalah tersebut diusulkan proses *split* pada *EBD* dengan dua fase. Adapun contoh data redundansi data biner dengan kelas yang berbeda dapat dilihat pada Tabel 3.4. Pada Tabel 3.4 diasumsikan terdapat dataset kesalahan perangkat lunak dengan enam fitur yang terdiri dari *loc*, *iv(g)*, *e*, *b*, *branchCount*, dan *faulty*. Data nomor 1 dan 4 menunjukkan redundansi data biner dengan kelas yang berbeda. Sedangkan proses penanganan redundansi dengan kelas yang berbeda ada di Gambar 3.8.



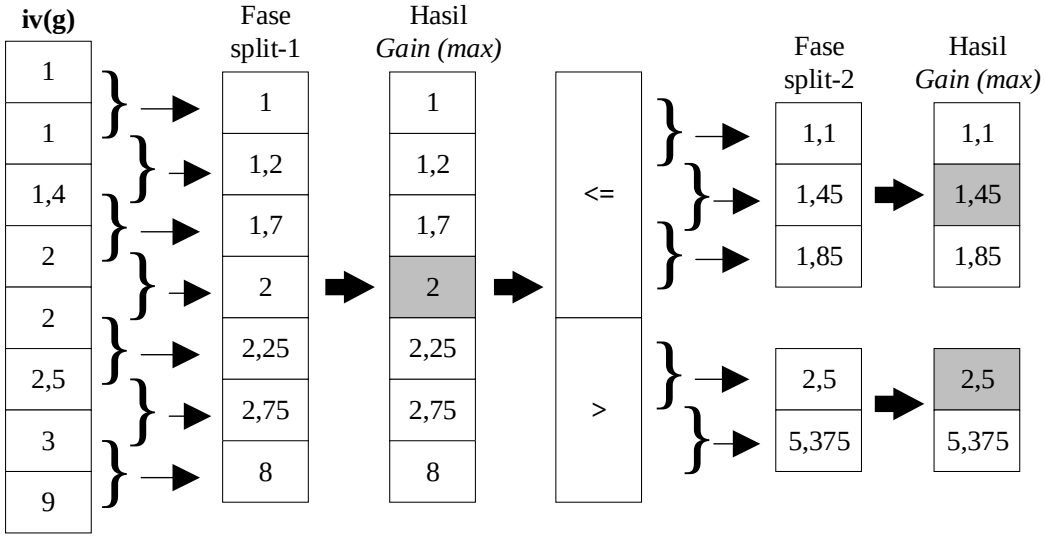
Gambar 3.8 Ilustrasi Proses *Split EBD* dengan Dua Fase

Tabel 3.4 Contoh Data Biner dengan Kelas yang Berbeda

No	loc	iv(g)	e	b	branchCount	faulty
1	**1**	**0**	**1**	**1**	**1**	**F**
2	0	0	0	1	0	T
3	1	0	1	0	1	F

No	loc	iv(g)	e	b	branchCount	faulty
4	**1**	**0**	**1**	**1**	**1**	**T**
5	1	1	1	1	1	T

Pada Gambar 3.8 dijelaskan bahwa hasil diskritisasi data numerik dijadikan biner dua bit. *Input* yang digunakan adalah data *split* pada Gambar 3.6. Usulan tersebut dilakukan untuk mencegah nilai biner yang sama dengan kelas berbeda. Daftar data *split* yang telah dihitung di proses sebelumnya, akan dijadikan sebagai *input* untuk melakukan *split* pada hasil *split* pertama. Jadi, data akan terbagi dua berdasarkan nilai yang kurang dari sama dengan (<=) dan data dengan nilai yang lebih dari (>). Adapun contoh proses *split* dengan dua fase dapat dilihat pada Gambar 3.9.



Gambar 3.9 Contoh Proses *Split* dengan Dua Fase

Pada Gambar 3.9 diasumsikan bahwa proses *split* dilakukan pada fitur *iv(g)*. Fitur *iv(g)* memilki delapan data yang kemudian di-*split* menjadi tujuh data. Dikarenakan proses *EBD* menggunakan nilai *maximum gain*, maka nilai *split* dengan *gain* tertinggi akan dipilih untuk mengubah data numerik menjadi data biner. Kotak yang berwarna abu-abu pada Gambar 3.9 menandakan bahwa nilai *split* tersebut memiliki nilai *gain* tertinggi. Diasumsikan pula bahwa hasil diskritisasi biner terdapat redundansi data. Untuk melakukan fase *split* yang kedua, maka hasil *gain(max)* pada fase pertama dijadikan acuan untuk memulai *split* pada fase kedua dengan pemisah *operator* kurang dari sama dengan (<=) dan *operator* lebih dari (>). Nilai yang diperoleh dari fase *split* yang kedua akan dipilih sebagai

kriteria untuk mengonversi data numerik berdasarkan nilai *gain* tertinggi. Contoh hasil *EBD* 2 fase untuk menangani redundansi data biner dengan kelas yang berbeda (berdasarkan Tabel 3.4) dapat dilihat di Tabel 3.5.

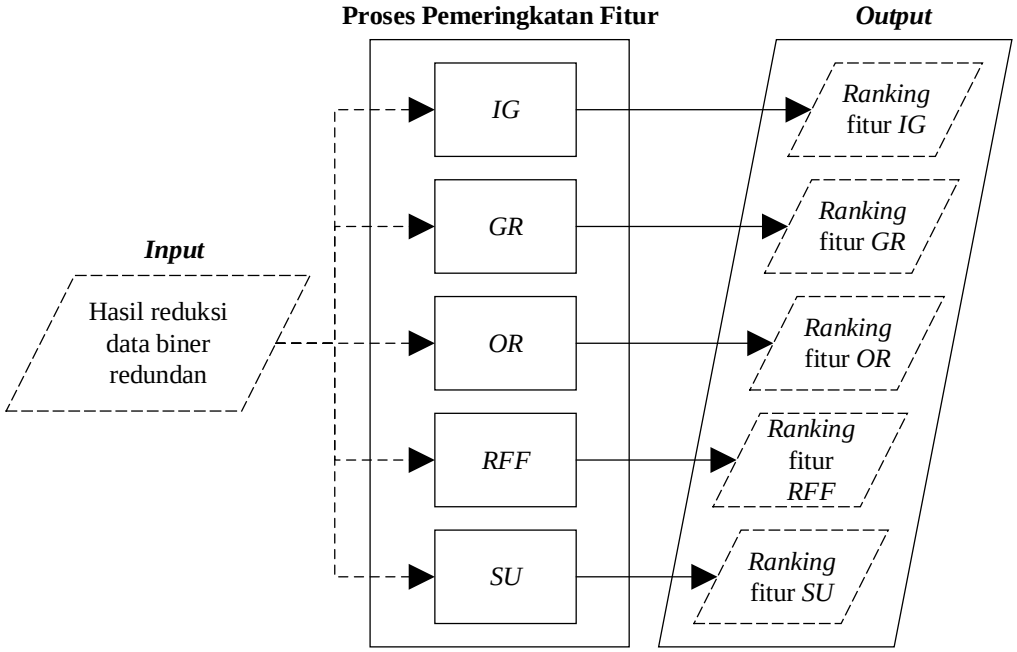
Tabel 3.5 Contoh Hasil *EBD* 2 fase

No	loc	iv(g)	e	b	branchCount	faulty
1	1**0**	0**0**	1**1**	1**0**	1**0**	**F**
2	00	01	00	10	00	T
3	11	00	11	01	10	F
4	1**0**	0**0**	1**0**	1**1**	1**1**	**T**
5	10	10	10	10	10	T

Tabel 3.5 adalah hasil diskritisasi *EBD* 2 fase, yaitu data biner dengan 2 digit. Jika pada Tabel 3.4 terdapat redundansi data biner (satu digit) dengan kelas yang berbeda (baris ke 1 dan 4), maka dengan melakukan proses *EBD* 2 fase tidak ada lagi redundansi data biner dengan kelas yang berbeda.

3.2.5. Proses Seleksi Fitur

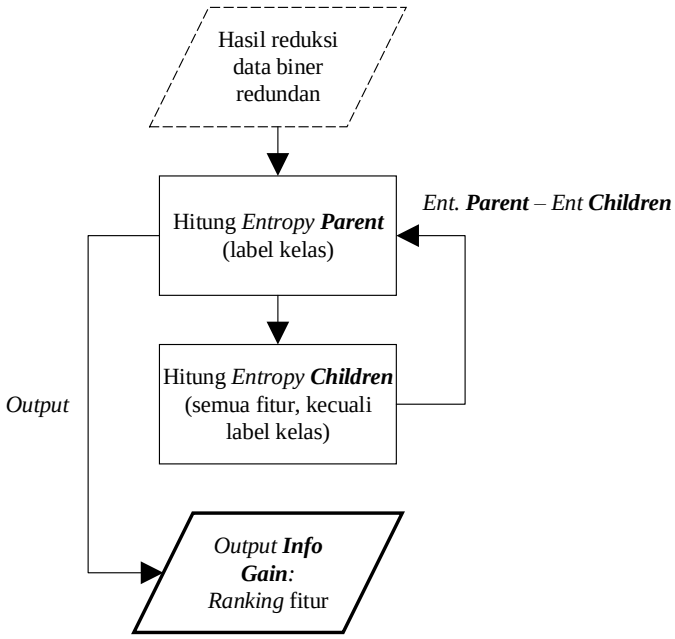
Setelah dataset di-diskritisasi, maka fitur pada dataset akan dilakukan penilaian *(scoring)* berdasarkan percobaan lima metode seleksi fitur yaitu *IG*, *GR*, *OR*, *RFF*, dan *SU*. *Output* dari proses seleksi fitur adalah peringkat fitur berdasarkan hasil perhitungan *score* fitur. Setiap metode seleksi fitur dapat menghasilkan perhitungan peringkat fitur yang berbeda. Adapun urutan langkah seleksi fitur pada tahap ini dapat dilihat pada Gambar 3.10.



Gambar 3.10 Urutan Proses Pemeringkatan Fitur

A. Proses Pemeringkatan Fitur pada *Information Gain (IG)*

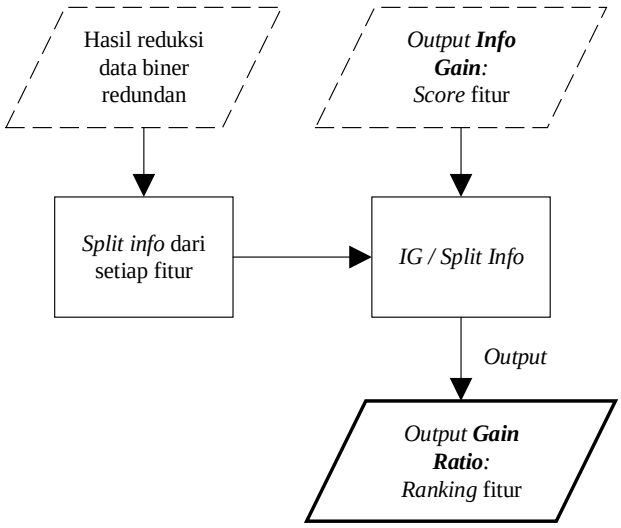
IG memanfaatkan hasil perhitungan *entropy* pada dataset dan *entropy* setiap fiturnya dari data diskrit hasil proses *EBD*. *Entopy* pada satu dataset disebut *entropy* parent, sedangkan *entropy* setiap fitur pada satu dataset disebut *entropy* children. Berdasarkan Gambar 3.11, *ranking* fitur diperoleh dari *score* fitur. Adapun *score* fitur didapat dari hasil pengurangan nilai *entropy parent* dengan *entropy children*. Semakin besar *score* fitur, maka semakin tinggi pula *ranking* fitur tersebut.



Gambar 3.11 Urutan Proses Pemeringkatan Fitur dengan *IG*

B. Proses Pemeringkatan Fitur pada *Gain Ratio (GR)*

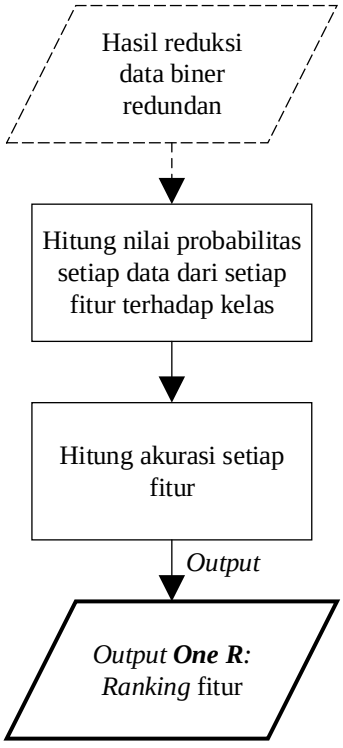
GR tidak hanya menggunakan *output* proses *EBD*, tetapi juga *score* dari *ranking* fitur *IG*. Berdasarkan Gambar 3.12, langkah pertama *GR* adalah menghitung nilai *split* info dari hasil proses *EBD*. Setelah nilai *split* info diketahui, maka *score IG* dari setiap fitur dibagi dengan masing-masing *split* info tiap fitur. Hasil pembagian tersebut adalah *score GR*. Sama halnya dengan *IG*, semakin tinggi *score* yang diperoleh semakin tinggi pula *ranking* fitur tersebut.



Gambar 3.12 Urutan Proses Pemeringkatan Fitur dengan *GR*

C. Proses Pemeringkatan Fitur pada *One-R (OR)*

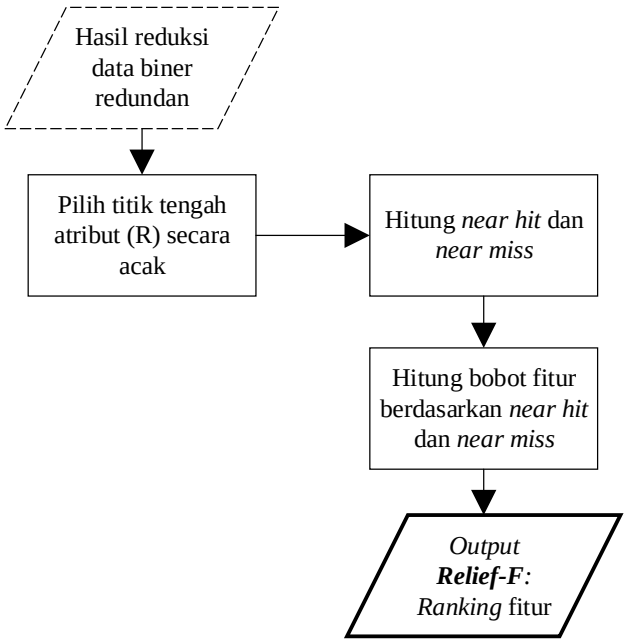
OR memanfaatkan nilai probabilitas setiap fitur untuk mememberikan *score* pada setiap fitur. Berdasarkan Gambar 3.13, pada proses OR, sebelum *score* ditentukan, maka perlu untuk menghitung nilai akurasi dari setiap fitur terlebih dahulu. Nilai akurasi fitur tersebut dijadikan sebagai *score* fitur. Nilai *score* yang tinggi berbanding lurus dengan *ranking* fitur.



Gambar 3.13 Urutan Proses Pemeringkatan Fitur dengan *OR*

D. Proses Pemeringkatan Fitur pada *Relief-F (RFF)*

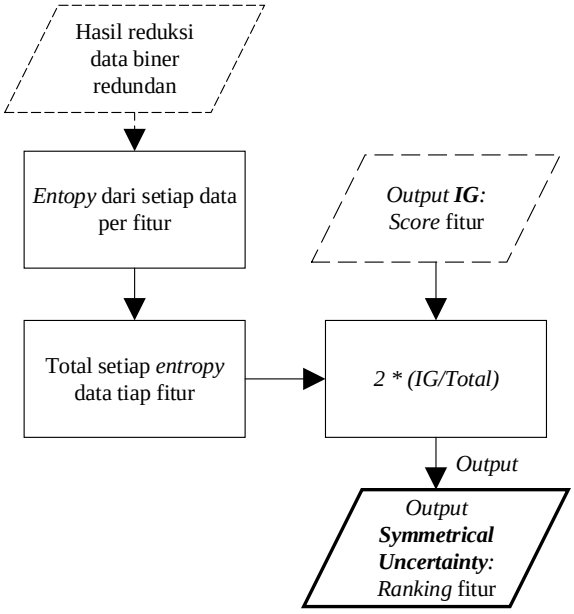
Untuk mendapatkan *score* dari setiap fitur, *RFF* memanfaatkan nilai *near miss* dan *near hit*. Berdasarkan Gambar 3.14, pada proses *RFF*, sebelum mencari nilai *near miss* dan *near hit*, perlu ditentukan terlebih dahulu nilai tengahnya *(R)*. Penentuan nilai tengah dilakukan secara acak. Setelah nilai *R* ditentukan, nilai *near miss* dan nilai *near hit* diketahui, maka *score* fitur dapat dihitung. *Output* pada proses ini adalah urutan *ranking* fitur berdasarkan perhitungan *score*-nya. Urutan proses *RFF* dapat dilihat pada Gambar 3.14.



Gambar 3.14 Urutan Proses Pemeringkatan Fitur dengan *RFF*

E. Proses Pemeringkatan Fitur pada *Symmetrical Uncertainty (SU)*

Sama halnya dengan *GR*, *SU* juga memanfaatkan *output score* dari *IG*. Namun, berdasarkan Gambar 3.15, sebelum menggunakan *score* fitur dari *IG*, *SU* perlu untuk mengetahui nilai total *entropy* dari setiap pilihan data per fitur. Setelah itu *score IG* dibagi dengan nilai total tersebut dan dikalikan dua. Hasil perhitungan tersebut adalah *score* fitur dari *SU*.



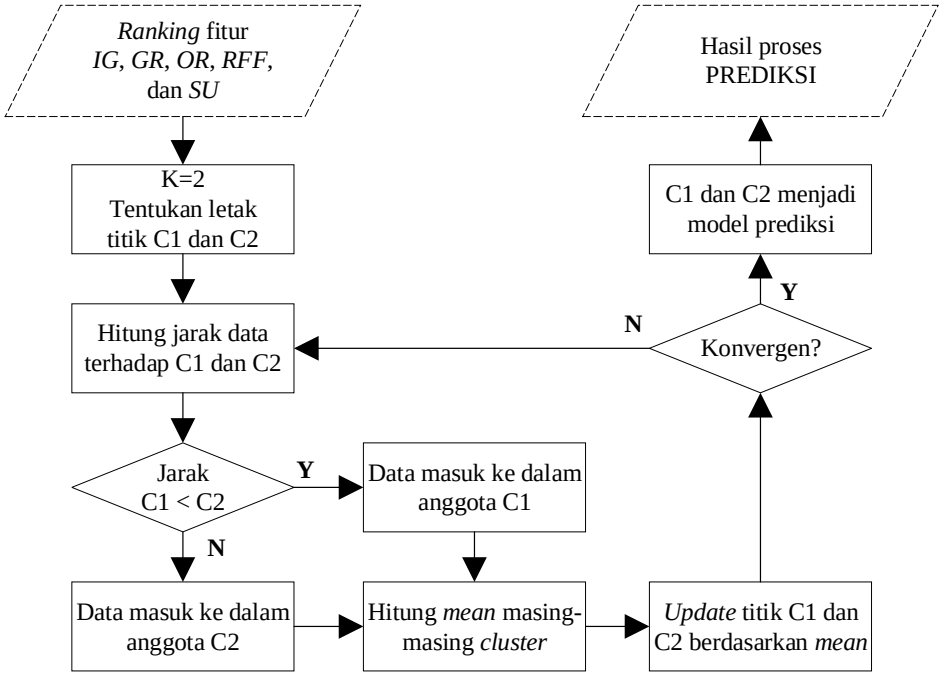
Gambar 3.15 Urutan Proses Pemeringkatan Fitur dengan *SU*

3.2.6. Prediksi Kesalahan Perangkat Lunak

Setelah semua data dilakukan pra-proses, maka selanjutnya data akan dijadikan sebagai input untuk model prediksi. Prediksi kesalahan perangkat lunak dilakukan dengan menggunakan metode *cluster-based classification (CBC)* (Singh & Verma 2014). Pada dasarnya, *clustering* adalah metode untuk mengelompokkan data berdasarkan karakteristik yang sama ke dalam kelompok yang sama dan mengelompokkan data dengan karakteristik yang berbeda ke dalam kelompok yang berbeda pula. Kesamaan karakteristik data dihitung menggunakan perhitungan jarak data ke masing-masing titik *cluster*. Sebuah data akan menjadi anggota kelompok titik *cluster (cluster-center)* tertentu jika mempunyai jarak paling minimum (dekat). Setelah kumpulan data menjadi sebuah *cluster*, maka selanjutnya akan dicari *cluster-center* baru dengan cara mencari nilai tengah *(mean)* dari anggota *cluster* yang bersangkutan, hingga konvergen. Pada penelitian ini konvergen dideskripsikan sebagai kondisi di mana tidak ada perubahan anggota data di *cluster-center* baru terhadap anggota data di *cluster-center* sebelumnya.

Pada kasus ini, jumlah *cluster-center (k)* dibagi menjadi dua. Penentuan nilai $k = 2$ didasarkan oleh jumlah kelas pada dataset kesalahan perangkat lunak, yaitu *faulty (true)* dan *non-faulty (false).* Artinya data akan dibagi berdasarkan dua

kelompok tersebut. Adapun urutan proses prediksi kesalahan perangkat lunak menggunakan *CBC* dapat dilihat pada Gambar 3.16.



Gambar 3.16 Proses Prediksi Kesalahan Perangat Lunak dengan *CBC*

Pada Gambar 3.16 dijelaskan bahwa data C1 dan C2 ditentukan berdasarkan kelas *true* dan *false*. Jika C1 mewakili kelas *true*, maka C2 harus mewakili kelas *false*, dan begitu juga sebaliknya. Dikarenakan data hasil diskritisasi adalah data biner, maka perhitungan jarak yang digunakan pada penelitian ini adalah *hamming distance* (sub-bab 2.6). Ketika sebuah data cenderung lebih dekat jaraknya terhadap C1, maka data tersebut masuk ke dalam anggota C1, dan sebaliknya. *Cluster-center* pada setiap *cluster* diperbarui berdasarkan *mean* dari masing-masing *cluster*. Apabila anggota *cluster* sudah konvergen, maka titik *cluster-center* tersebut akan dijadikan acuan model prediksi dalam menentukan keanggotaan data, *false* atau *true*.

Namun, dikarenakan *hamming distance* hanya mendukung perhitungan data dengan tipe data biner, maka hasil *mean* yang umumnya bertipe data *float/double* perlu dilakukan pembulatan hingga menjadi data biner kembali. Dikarenakan *output EBD* 2 fase adalah dua digit biner, maka *range* pembulatan *mean* dimulai dari angka 0 hingga 3. Adapun *rule* pembulatan yang digunakan pada penelitian ini dapat dilihat pada Tabel 3.6.

Tabel 3.6 *Rule* Pembulatan Hasil *Mean*

No	*Rule*	Pembulatan (Desimal)	Biner
1	$0 < x \leq 0{,}5$	0	00
2	$0{,}5 < x \leq 1{,}5$	1	01
3	$1{,}5 < x \leq 2{,}5$	2	10
4	$x > 2{,}5$	3	11

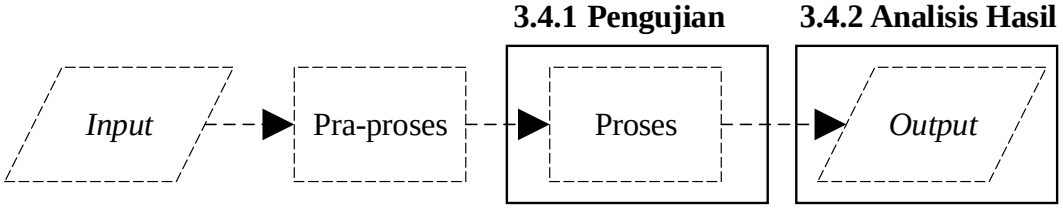
Pada Tabel 3.6, variabel *x* menunjukkan nilai *mean* dari perhitungan nilai biner yang telah didesimalkan sebelumnya. Dikarenakan jumlah *digit* biner adalah 2, maka nilai maksimal angka desimal yang dapat dihasilkan yaitu 3.

3.3. Implementasi Metode

Pada penelitian ini metode yang diusulkan khususnya metode *EBD* dan *CBC* akan diimplementasikan menggunakan *IDE (Integrated Development Environment)* MATLAB versi R2016a. Sedangkan usulan lima metode seleksi fitur *IG*, *GR*, *OR*, *RFF*, dan *SU* akan diimplementasikan dengan menggunakan kakas bantu WEKA versi 3.8.

3.4. Rancangan Pengujian dan Analisis Hasil

Pada tahap ini, dilakukan pengujian dan analisis hasil dari usulan metode. Proses ini terdiri dari dua sub-proses yaitu rancangan pengujian dan rancangan analisis hasil. Secara umum, proses ini dapat dilihat pada Gambar 3.17.

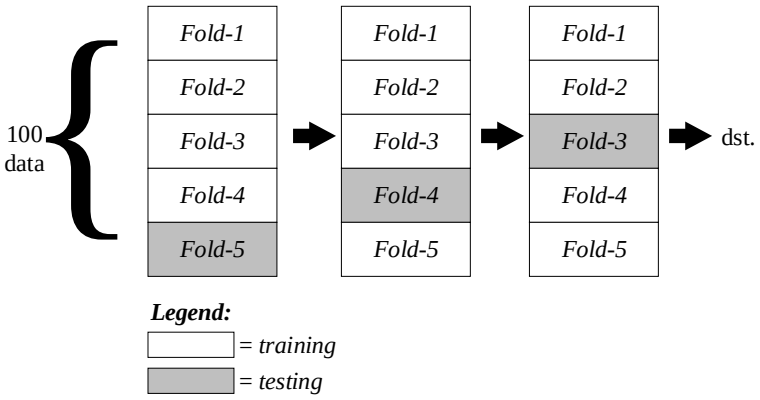


Gambar 3.17 Proses Pengujian dan Analisis Hasil

3.4.1. Rancangan Pengujian

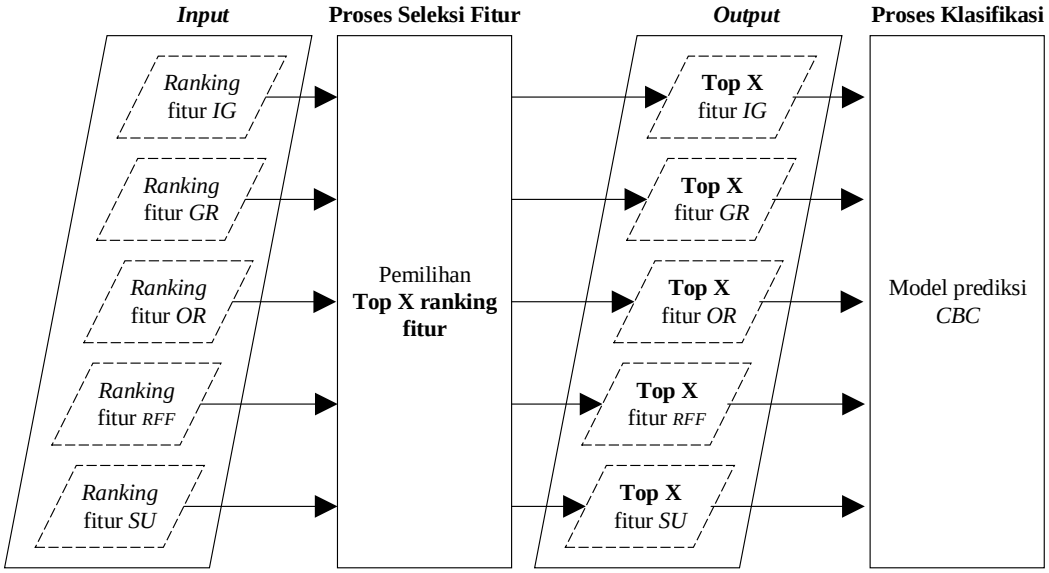
Metode yang paling umum digunakan untuk mendeteksi kasus kesalahan pada model prediksi adalah *k-fold cross validation.* Satu putaran *cross-validation* membagi data ke dalam dua kategori data yaitu data *training* dan data *testing*. Jika terdapat 100 data dan *fold* terbagi menjadi 5-*fold*, maka 4-*fold* akan dijadikan sebagai data *training* dan 1-*fold* sebagai data *testing*. Perulangan dilakukan hingga

semua *fold* data pernah dijadikan sebagai data *testing* (Singh et al. 2016). Adapun ilustrasi proses *k-fold cross validation* dapat dilihat pada Gambar 3.18.



Gambar 3.18 Proses *K-Fold Cross Validation*

Berdasarkan Gambar 3.18, diasumsikan terdapat 100 data yang akan dibagi ke dalam 5 fold. Jika dibagi menjadi 5 fold, maka setiap fold akan terdiri dari sekitar 20 data. Setelah setiap metode seleksi fitur menghasilkan *ranking* fitur, maka fitur-fitur tersebut dipilih. Pemilihan fitur dilakukan untuk menentukan fitur mana saja yang dianggap paling relevan dengan *output* model prediksi. Semakin tinggi *ranking* fitur, maka fitur tersebut semakin relevan. Sejumlah *top X ranking* fitur nantinya akan dipilih secara iteratif untuk dijadikan *input* model prediksi *CBC*. Adapun urutan proses pemilihan *top X ranking* fitur beserta *output*-nya dapat dilihat pada Gambar 3.19.



Gambar 3.19 Proses Pemilihan *Top X Ranking* Fitur

Pada Gambar 3.19, kelima versi peringkat fitur akan dilakukan pemilihan *Top X* fitur secara iteratif untuk menemukan kombinasi fitur yang paling relevan dengan *output* prediksi. Semakin relevan fitur yang digunakan, semakin tinggi pula nilai *output* prediksi.

3.4.2. Rancangan Analisis Hasil

Akurasi dan kinerja dari model prediksi untuk masalah dua kelas yaitu yang positif terdapat kesalahan perangkat lunak *(true)* dan yang tidak *(false)*, umumnya dievaluasi dengan menggunakan matrik *confusion* (Singh & Verma 2014). Sebuah matrik *confusion* berisi informasi tentang klasifikasi aktual dan prediksi yang dilakukan oleh sistem klasifikasi. Penelitian ini menggunakan ukuran kinerja prediksi yang umum digunakan yaitu probabilitas *detection (pd)*, probabilitas *false alarm (pf)*, dan *balance (bal)* untuk mengevaluasi dan membandingkan model prediksi. Dikarenakan tidak imbangnya perbandingan kelas pada dataset kesalahan perangkat lunak, maka penggunaan parameter akurasi tidak dapat digunakan (Catal 2012). Adapun matrik *confusion* pada penelitian ini dapat dilihat pada Tabel 2.5 di BAB II.

Prediksi *false alarm* (pf), sebaiknya bernilai nol, artinya bahwa prediktor seharusnya tidak pernah memprediksi modul perangkat lunak yang tidak salah *(false)* sebagai modul yang salah *(true)*. Secara umum, peningkatan nilai *pd* juga akan meningkatkan nilai *pf* karena modul lebih sering mencapai hal yang ideal. Pemilihan nilai *pd* dan *pf* yang terbaik, ditentukan oleh nilai *balance*. Semakin tinggi nilai *balance*, maka nilai *pd* akan semakin mendekati 1 dan nilai *pf* akan semakin mendekati nilai 0 (Menzies et al. 2007). Untuk mendapatkan nilai pengukuran, maka rumus yang digunakan dapat dilihat di rumus 2.11, 2.12, dan 2.13.

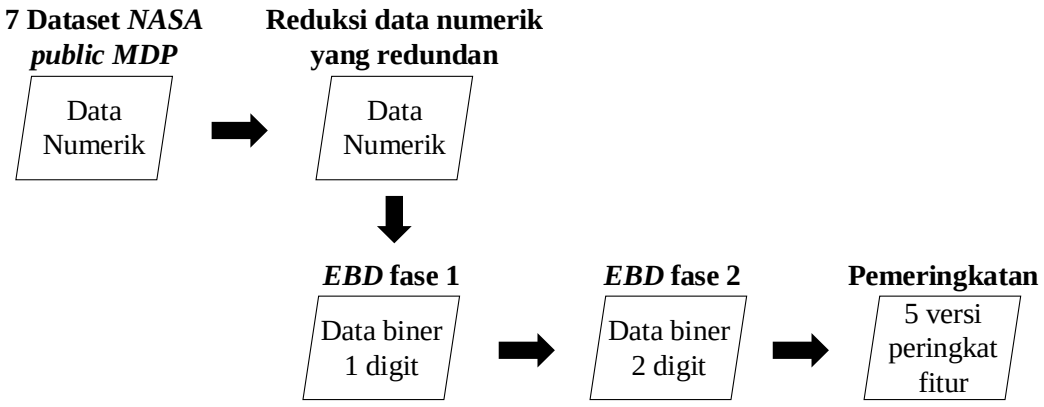
[Halaman ini sengaja dikosongkan]

BAB 4
UJI COBA DAN EVALUASI

Bab ini dijelaskan tentang uji coba dan evaluasi pada usulan metode yang telah dilakukan. Terdapat tiga sub pokok bahasan, yaitu implementasi penelitian, perancangan uji coba, dan analisis hasil.

4.1. Implementasi Penelitian

Pada bagian ini dipaparkan tentang proses diskritisasi tujuh dataset *NASA public MDP* yang bertipe data numerik hingga menjadi tipe data biner dua digit. Setelah itu, setiap fitur pada dataset akan dilakukan pemeringkatan menggunakan lima metode seleksi fitur yang berbeda. Adapun metode diskritisasi yang digunakan pada penelitian ini adalah *EBD*. Diskritisasi dilakukan untuk dapat meningkatkan akurasi prediksi kesalahan perangkat lunak dan meningkatkan efisiensi pada proses komputasi (Singh & Verma 2014) (Singh & Vyas 2014). Sedangkan kelima metode seleksi fitur yang digunakan untuk menentukan peringkat fitur adalah *GR*, *IG*, *OR*, *RFF*, dan *SU*. Ketika data numerik pada dataset sudah ditransformasikan menjadi data biner dua digit dan fiturnya telah diurutkan berdasarkan masing-masing metode seleksi fitur, maka skenario pengujian siap untuk dilakukan. Adapun tahapan yang perlu dilakukan sebelum skenario pengujian dimulai yaitu dapat dilihat pada Gambar 4.1.



Gambar 4.1 Tahapan Implementasi Penelitian

4.1.1. Implementasi Reduksi Data Numerik yang Redundan dengan Kelas yang Sama

Dari ketujuh dataset *NASA public MDP*, PC2 adalah dataset yang terdapat redundansi data numerik terbanyak, yaitu 74,84% redundasi data dengan kelas yang sama. Namun, pada semua dataset tidak terdapat redundansi data numerik dengan kelas yang berbeda. Adapun rincian redundansi pada tujuh dataset *NASA public MDP* disajikan pada Tabel 4.1.

Tabel 4.1 Rincian Redundansi Data Numerik

No	Dataset	Jumlah fitur	Jumlah data	Jumlah redundansi	Jumlah data setelah reduksi	% redundansi
1	CM1	22	498	56	442	11,24
2	KC3	40	458	132	326	28,82
3	MW1	38	403	21	382	5,21
4	PC1	22	1109	155	954	13,97
5	PC2	37	5589	4183	1406	**74,84**
6	PC3	38	1563	124	1439	7,93
7	PC4	38	1458	114	1344	7,82

4.1.2. Implementasi *EBD* Fase 1

Pada tahap ini, hasil reduksi data numerik yang redundan pada proses Sub-bab 4.1.1, akan dijadikan sebagai *input EBD* fase pertama. Adapun langkah-langkah yang dilakukan pada tahap ini dapat dilihat pada Sub-bab 3.2.2. Semua fitur pada dataset *NASA public MDP* akan ditransformasikan menjadi data biner satu digit yaitu 0 atau 1. Namun, dikarenakan adanya *gap* interval yang tinggi pada dataset, maka akan muncul kemungkinan redundansi data biner dengan kelas yang berbeda. Contoh redundansi data biner dengan kelas yang berbeda dapat dilihat pada Sub-bab 3.2.4. Redundansi data biner dengan kelas yang berbeda dapat membuat sistem menjadi tidak konsisten dalam melakukan prediksi.

Sebagai contoh, pada Sub-bab ini akan disajikan penentuan *rule* untuk mengubah data numerik menjadi data biner satu digit pada dataset CM1 yang terdiri dari 21 fitur. *Rule EBD* fase 1 di dataset CM1 dapat dilihat di Tabel 4.2.

Tabel 4.2 Proses *EBD* Fase 1 di CM1

No	Fitur CM1	*Best Split* Fase 1	*Rule* Fase 1	Biner 1 digit
1	LOC	27	<= 27	0
			> 27	1
2	v(g)	2	<= 2	0
			> 2	1
3	ev(g)	27	<= 27	0
			> 27	1

No	Fitur CM1	*Best Split* Fase 1	*Rule* Fase 1	Biner 1 digit
4	iv(g)	2	<= 2	0
			> 2	1
5	N	103	<= 103	0
			> 103	1
6	V	554,775	<= 554,775	0
			> 554,775	1
7	L	0,08	<= 0,08	0
			> 0,08	1
8	D	11,85	<= 11,85	0
			> 11,85	1
9	I	30,665	<= 30,665	0
			> 30,665	1
10	E	4856,745	<= 4856,745	0
			> 4856,745	1
11	B	0,18	<= 0,18	0
			> 0,18	1
12	T	269,815	<= 269,815	0
			> 269,815	1
13	SLOC	0	<= 0	0
			> 0	1
14	CLOC	6	<= 6	0
			> 6	1
15	BLOC	7	<= 7	0
			> 7	1
16	C&SLOC	0	<= 0	0
			> 0	1
17	n1	18	<= 18	0
			> 18	1
18	n2	20	<= 20	0
			> 20	1
19	N1	52	<= 52	0
			> 52	1
20	N2	49,5	<= 49,5	0
			> 49,5	1
21	Branch_C	4	<= 4	0
			> 4	1

Berdasarkan implementasi *EBD* fase pertama, hasil menunjukkan bahwa terdapat duplikasi data biner dengan kelas yang berbeda pada setiap dataset. Rata-rata jumlah duplikasi data biner dengan kelas yang berbeda adalah 17,85 data. Jumlah duplikasi tersebut diketahui dengan melakukan *distinct* pada semua fitur termasuk kelas dan proses *distinct* pada semua fitur tanpa kelas. Rincian hasil *EBD* fase pertama dapat dilihat pada Tabel 4.3.

Tabel 4.3 Rincian Redundansi Data Biner dengan Kelas Berbeda di *EBD* Fase Pertama

No	Dataset	Data numerik			Data biner hasil EBD fase 1				
		Total data	Kelas		Total data *distinct*		Redundansi data biner kelas berbeda	Kelas	
			true	*false*	tanpa kelas	dengan kelas		*true*	*false*
1	CM1	442	48	394	155	167	**12**	26	141
2	KC3	326	43	283	200	210	**10**	38	172
3	MW1	382	31	351	149	159	**10**	22	137
4	PC1	954	70	884	203	227	**24**	36	191
5	PC2	1406	23	1383	493	497	**4**	21	476
6	PC3	1439	153	1286	887	923	**36**	131	792
7	PC4	1344	177	1167	832	861	**29**	104	757
						Rata-rata	**17,85**		

4.1.3. Implementasi *EBD* Fase 2

Tahap ini dilakukan untuk mengurangi jumlah duplikasi data biner dengan kelas yang berbeda dari tahap *EBD* fase pertama dengan menambah satu digit biner menjadi dua digit biner (00, 01, 10, dan 11). Setiap fase pada *EBD* dilakukan secara iteratif hingga tidak ada lagi duplikasi data dengan kelas yang berbeda. Namun, pada penelitian ini *EBD* hanya dilakukan hingga dua fase saja. Sebagai contoh, pada Sub-bab ini akan disajikan penentuan *rule* untuk mengubah data numerik menjadi data biner dua digit pada dataset CM1 yang terdiri dari 21 fitur. *Rule EBD* fase 2 di dataset CM1 dapat dilihat di Tabel 4.4.

Tabel 4.4 Proses *EBD* Fase 2 di CM1

No	Fitur CM1	*Best Split* Fase 1	*Rule* Fase 1	Biner 1 digit	*Best Split* Fase 2	*Rule* Fase 2	Biner 2 digit
1	LOC	27	<= 27	0	17,25	<= 17,25	00
						> 17,25	01
			> 27	1	70	<= 70	10
						> 70	11
2	v(g)	2	<= 2	0	1,45	<= 1,45	00
						> 1,45	01
			> 2	1	17,25	<= 17,25	10
						> 17,25	11
3	ev(g)	27	<= 27	0	1,7	<= 1,7	00
						> 1,7	01
			> 27	1	28,5	<= 28,5	10
						> 28,5	11
4	iv(g)	2	<= 2	0	1,45	<= 1,45	00
						> 1,45	01
			> 2	1	15,25	<= 15,25	10
						> 15,25	11
5	N	103	<= 103	0	23,25	<= 23,25	00
						> 23,25	01
			> 103	1	498,5	<= 498,5	10
						> 498,5	11
6	V	554,775	<= 554,775	0	102,88	<= 102,88	00
						> 102,88	01
			> 554,775	1	590,5025	<= 590,5025	10
						> 590,5025	11
7	L	0,08	<= 0,08	0	0,0525	<= 0,0525	00
						> 0,0525	01
			> 0,08	1	0,2025	<= 0,2025	10
						> 0,2025	11

No	Fitur CM1	*Best Split* Fase 1	*Rule* Fase 1	Biner 1 digit	*Best Split* Fase 2	*Rule* Fase 2	Biner 2 digit
8	D	11,85	<= 11,85	0	4,89	<= 4,89	00
						> 4,89	01
			> 11,85	1	27,895	<= 27,895	10
						> 27,895	11
9	I	30,665	<= 30,665	0	17,7125	<= 17,7125	00
						> 17,7125	01
			> 30,665	1	160,68	<= 160,68	10
						> 160,68	11
10	E	4856,745	<= 4856,745	0	490,9425	<= 490,9425	00
						> 490,9425	01
			> 4856,745	1	33678,78	<= 33678,78	10
						> 336678,78	11
11	B	0,18	<= 0,18	0	0,0225	<= 0,0225	00
						> 0,0225	01
			> 0,18	1	0,1925	<= 0,1925	10
						> 0,1925	11
12	T	269,815	<= 269,815	0	27,275	<= 27,275	00
						> 27,275	01
			> 269,815	1	1871,045	<= 1871,045	10
						> 1871,045	11
13	SLOC	0	<= 0	0	0	<= 0	00
						> 0	01
			> 0	1	5,25	<= 5,25	10
						> 5,25	11
14	CLOC	6	<= 6	0	0,25	<= 0,25	00
						> 0,25	01
			> 6	1	58,25	<= 58,25	10
						> 58,25	11
15	BLOC	7	<= 7	0	6,25	<= 6,25	00
						> 6,25	01
			> 7	1	26,25	<= 26,25	10
						> 26,25	11
16	C&SLOC	0	<= 0	0	0	<= 0	00
						> 0	01
			> 0	1	1	<= 1	10
						> 1	11
17	n1	18	<= 18	0	7,25	<= 7,25	00
						> 7,25	01
			> 18	1	36,25	<= 36,25	10
						> 36,25	11
18	n2	20	<= 20	0	7,25	<= 7,25	00
						> 7,25	01
			> 20	1	110,5	<= 110,5	10
						> 110,5	11
19	N1	52	<= 52	0	1,05	<= 1,05	00
						> 1,05	01
			> 52	1	53,25	<= 53,25	10
						> 53,25	11
20	N2	49,5	<= 49,5	0	9,25	<= 9,25	00
						> 9,25	01
			> 49,5	1	217,25	<= 217,25	10
						> 217,25	11
21	Branch_C	4	<= 4	0	3,25	<= 3,25	00
						> 3,25	01
			> 4	1	31,25	<= 31,25	10
						> 31,25	11

Berdasarkan hasil implementasi, diperoleh jumlah rata-rata duplikasi data dengan kelas berbeda yang lebih sedikit jika dibandingkan dengan *EBD* fase 1, yaitu dari 17,85 menjadi 7,28. Adapun rincian hasil pada *EBD* fase kedua, disajikan pada Tabel 4.5.

Tabel 4.5 Rincian Redundansi Data Biner dengan Kelas Berbeda di *EBD* Fase Kedua

No	Dataset	Data numerik			Data biner hasil EBD fase 2				
		Total data	Kelas		Total data *distinct*		Redundansi data kelas berbeda	Kelas	
			true	false	tanpa kelas	dengan kelas		true	false
1	CM1	442	48	394	338	345	**7**	46	299
2	KC3	326	43	283	287	288	**1**	43	245
3	MW1	382	31	351	327	332	**5**	31	301
4	PC1	954	70	884	560	579	**19**	58	521
5	PC2	1406	23	1383	859	862	**3**	22	840
6	PC3	1439	153	1286	1274	1281	**7**	151	1130
7	PC4	1344	177	1167	1219	1228	**9**	164	1064
						Rata-rata	**7,28**		

Jika dilakukan perbandingan jumlah duplikasi data pada *EBD* fase pertama dan kedua, maka *EBD* pada fase kedua terdapat lebih sedikit jumlah duplikasi. Perbandingan jumlah duplikasi dalam persentase dapat dilihat pada Tabel 4.6.

Tabel 4.6 Perbandingan Jumlah Redundansi *EBD* Fase 1 dan Fase 2

No	Dataset	*EBD* Fase 1			*EBD* Fase 2		
		Data	Duplikasi	**%**	Data	Duplikasi	**%**
1	CM1	167	12	**7,18**	345	7	**2,02**
2	KC3	210	10	**4,76**	288	1	**0,34**
3	MW1	159	10	**6,28**	332	5	**1,50**
4	PC1	227	24	**10,5**	579	19	**3,28**
5	PC2	497	4	**0,80**	862	3	**0,34**
6	PC3	923	36	**3,90**	1281	7	**0,54**
7	PC4	861	29	**3,36**	1228	9	**0,73**

4.1.4. Implementasi Pemeringkatan Fitur

Setelah *EBD* dua fase, maka tahap selanjutnya yang perlu dilakukan sebelum memasuki tahap uji coba adalah pemeringkatan fitur. *Output* pada tahap ini adalah lima versi *ranking* fitur yang didasarkan pada lima metode seleksi fitur yaitu *GR*, *IG*, *OR*, *RFF*, dan *SU*. Adapun tahapan-tahapan pada proses pemeringkatan fitur dapat dilihat di Sub-bab 3.2.5. Proses implementasi pemeringkatan fitur pada penelitian ini menggunakan kakas bantu WEKA versi 3.8.

A. Pemeringkatan Fitur Pada CM1

Pada CM1 terdapat 345 data dan 21 fitur. Adapun persentase kelas *false* pada CM1 mencapai 86,6%, sedangkan kelas *true* hanya sebanyak 13,3%. Dari kelima versi *ranking* fitur pada CM1, jika diambil top 10 fitur, maka metode *GR* dan *IR* tedapat kesamaan anggota fitur paling banyak. Rincian urutan *ranking* fitur tertinggi hingga terendah pada CM1 dapat dilihat pada Tabel 4.7.

Tabel 4.7 *Ranking* Fitur Pada CM1

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
1	0,1582	16	0,08066	14	87,826	14	0,174	9	0,0736	11
2	0,0526	19	0,07159	11	87,826	6	0,1457	8	0,0708	14
3	0,0519	11	0,07005	6	87,536	11	0,1341	7	0,0695	19
4	0,0472	14	0,06683	1	87,246	19	0,1319	13	0,0666	6
5	0,0455	6	0,06643	18	86,667	7	0,1288	15	0,0602	18
6	0,0405	18	0,05912	9	86,667	5	0,1278	14	0,0567	17
7	0,0395	17	0,058	19	86,667	10	0,1269	18	0,0544	1
8	0,0367	9	0,05678	17	86,667	3	0,1038	2	0,0543	9
9	0,0354	1	0,05167	5	86,667	2	0,101	6	0,046	5
10	0,0307	5	0,04797	20	86,667	8	0,0904	20	0,0433	20
11	0,0291	20	0,04392	4	86,667	21	0,0875	11	0,0393	4
12	0,0263	4	0,04346	10	86,667	20	0,0849	4	0,0359	10
13	0,0238	15	0,04346	12	86,667	13	0,0811	19	0,0359	12
14	0,0234	12	0,04011	15	86,667	15	0,0761	17	0,0356	15
15	0,0234	10	0,02745	8	86,667	16	0,064	10	0,029	16
16	0,0201	21	0,02694	2	86,667	12	0,064	12	0,0281	21
17	0,0169	2	0,02636	21	86,667	1	0,0607	21	0,025	2
18	0,0155	8	0,02269	7	85,797	17	0,0555	5	0,0235	8
19	0,013	3	0,01692	13	85,797	9	0,0383	1	0,019	7
20	0,0125	7	0,01176	3	85,797	18	0,0359	16	0,0161	13
21	0,011	13	0,00905	16	84,928	4	0,024	3	0,016	3

B. Pemeringkatan Fitur Pada KC3

Pada KC3 terdapat 288 data dan 39 fitur. Adapun persentase kelas *false* pada KC3 mencapai 85,1%, sedangkan kelas *true* hanya sebanyak 14,9%. Dari kelima versi *ranking* fitur pada KC3, jika diambil top 20 fitur, maka metode *GR* dan *SU* tedapat kesamaan anggota fitur paling banyak. Rincian urutan *ranking* fitur tertinggi hingga terendah pada KC3 dapat dilihat pada Tabel 4.8.

Tabel 4.8 *Ranking* Fitur Pada KC3

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
1	0,14677	4	0,08883	24	87,1528	2	0.23103	24	0.1319	4
2	0,06075	16	0,08452	27	86,1111	28	0.2131	26	0.08305	26
3	0,06075	39	0,08431	2	86,1111	7	0.2131	22	0.08305	22
4	0,06024	22	0,08126	26	85,7639	12	0.19697	27	0.08304	2
5	0,06024	26	0,08126	22	85,0694	15	0.17804	33	0.08231	16
6	0,05926	2	0,07895	34	85,0694	39	0.17386	8	0.08231	39
7	0,05492	24	0,07759	39	85,0694	14	0.17261	23	0.07983	24
8	0,05306	27	0,07759	16	85,0694	17	0.16777	39	0.07681	27
9	0,0496	7	0,07739	33	85,0694	18	0.16777	16	0.0696	7
10	0,04807	14	0,07615	23	85,0694	16	0.1666	34	0.06774	1
11	0,04787	1	0,07433	37	85,0694	10	0.15367	35	0.06684	33
12	0,04594	20	0,07409	20	85,0694	38	0.13367	38	0.06672	20
13	0,04579	28	0,07369	35	85,0694	8	0.12845	20	0.06612	34
14	0,04532	33	0,07283	4	85,0694	3	0.12255	5	0.06434	23
15	0,04435	34	0,07142	3	85,0694	5	0.11912	36	0.06159	28
16	0,04329	23	0,07093	7	85,0694	6	0.10506	25	0.06126	37
17	0,04087	37	0,07041	1	85,0694	19	0.10354	32	0.05966	35
18	0,04055	38	0,06897	18	85,0694	20	0.10136	13	0.05738	13

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
19	0,03957	35	0,06521	11	85,0694	1	0.10047	37	0.05701	3
20	0,0394	13	0,06416	13	85,0694	29	0.09712	3	0.05641	38
21	0,03798	31	0,06198	31	85,0694	30	0.08664	21	0.05607	18
22	0,03764	3	0,06142	36	85,0694	31	0.08513	18	0.05534	31
23	0,03724	18	0,06073	21	85,0694	33	0.08066	4	0.05422	11
24	0,03628	11	0,05718	28	85,0694	35	0.07872	19	0.0535	14
25	0,0359	21	0,05636	38	85,0694	34	0.07593	11	0.05281	21
26	0,03555	36	0,05038	8	85,0694	27	0.07445	2	0.05259	36
27	0,03432	8	0,04477	32	85,0694	23	0.06379	17	0.04853	8
28	0,03308	6	0,04027	30	85,0694	24	0.06167	31	0.04473	32
29	0,03308	30	0,04027	6	85,0694	25	0.04392	12	0.04412	30
30	0,03237	5	0,04001	25	84,375	4	0.03925	7	0.04412	6
31	0,03213	32	0,03811	29	84,375	32	0.03519	15	0.04197	5
32	0,03187	9	0,03681	9	84,375	13	0.03356	28	0.04175	9
33	0,02851	29	0,03668	14	84,0278	37	0.02484	1	0.03919	29
34	0,02788	12	0,03628	5	84,0278	9	0.02074	14	0.03832	25
35	0,02703	25	0,03447	19	83,6806	21	0.01814	29	0.03499	12
36	0,02241	19	0,02856	12	83,6806	22	0.01376	10	0.03212	19
37	0,01901	15	0,02361	10	83,6806	26	0.00788	9	0.02409	15
38	0,01599	10	0,01999	15	83,6806	11	0.00147	6	0.02265	10
39	0,00154	17	0,00199	17	82,9861	36	0.00147	30	0.0021	17

C. Pemeringkatan Fitur Pada MW1

Pada MW1 terdapat 332 data dan 37 fitur. Adapun persentase kelas *false* pada MW1 mencapai 90,7%, sedangkan kelas *true* hanya sebanyak 9,3%. Dari kelima versi *ranking* fitur pada MW1, jika diambil top 10 fitur, maka metode *GR* dan *SU* tedapat kesamaan anggota fitur paling banyak. Rincian urutan *ranking* fitur tertinggi hingga terendah pada MW1 dapat dilihat pada Tabel 4.9.

Tabel 4.9 *Ranking* Fitur Pada MW1

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
1	0,07934	35	0,0946	29	91,2651	1	0,307227	25	0,11123	35
2	0,07222	29	0,09185	33	91,2651	3	0,300457	18	0,10765	29
3	0,06554	1	0,09176	25	91,2651	18	0,270811	24	0,0956	25
4	0,06309	13	0,08945	32	91,2651	25	0,270811	20	0,09496	13
5	0,06233	25	0,08784	18	90,6627	15	0,263867	21	0,09337	1
6	0,05911	14	0,08766	22	90,6627	17	0,258864	32	0,09033	18
7	0,05867	18	0,08589	13	90,6627	16	0,25529	22	0,08949	33
8	0,05807	37	0,08556	21	90,6627	13	0,225759	33	0,08806	32
9	0,05723	33	0,08326	35	90,6627	14	0,217415	3	0,08727	37
10	0,05647	32	0,08085	3	90,6627	12	0,206793	23	0,08629	22
11	0,05534	22	0,07855	37	90,6627	20	0,193719	36	0,08502	21
12	0,05467	21	0,07264	1	90,6627	6	0,189231	19	0,07875	3
13	0,05145	16	0,07209	5	90,6627	2	0,174856	5	0,07735	16
14	0,05035	3	0,07101	31	90,6627	7	0,160427	31	0,07334	31
15	0,0477	31	0,06971	16	90,6627	10	0,159419	34	0,07274	14
16	0,0475	11	0,06625	11	90,6627	9	0,140542	10	0,07192	11
17	0,04547	5	0,06119	24	90,6627	36	0,127627	29	0,07092	5
18	0,04297	27	0,06119	20	90,6627	37	0,125973	37	0,06478	27
19	0,04091	2	0,05889	27	90,6627	21	0,118675	13	0,06216	24
20	0,04028	6	0,0574	6	90,6627	33	0,107174	2	0,06216	20
21	0,04023	24	0,05724	9	90,6627	32	0,102323	8	0,06148	2
22	0,04023	20	0,05677	28	90,6627	34	0,10147	12	0,0613	6

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
23	0,03762	28	0,05533	2	90,6627	22	0,099411	16	0,05803	28
24	0,03734	9	0,04811	7	90,6627	35	0,091186	35	0,05781	9
25	0,03326	4	0,04232	14	90,6627	31	0,085603	11	0,05002	7
26	0,0326	7	0,0389	8	90,6627	30	0,064721	7	0,04443	8
27	0,02985	8	0,03507	12	90,6627	19	0,056358	4	0,04033	12
28	0,02716	12	0,02917	10	90,6627	27	0,048264	28	0,02985	26
29	0,02338	26	0,0251	36	90,6627	23	0,047653	6	0,02968	4
30	0,0185	30	0,02469	19	90,6627	24	0,043148	9	0,02761	10
31	0,01752	10	0,02198	23	90,6627	28	0,042488	27	0,02654	19
32	0,01747	19	0,01865	34	90,3614	4	0,029516	30	0,02613	36
33	0,01703	36	0,01849	26	90,3614	26	0,021531	1	0,02379	23
34	0,0157	23	0,012	4	90,0602	8	0,013986	26	0,01848	34
35	0,01187	34	0,00824	30	90,0602	29	0,008326	14	0,01845	30
36	0,006	17	0,00124	17	89,4578	5	0,000568	15	0,00379	17
37	0,00367	15	0,00121	15	89,4578	11	-0,006665	17	0,00311	15

D. Pemeringkatan Fitur Pada PC1

Pada PC1 terdapat 579 data dan 21 fitur. Adapun persentase kelas *false* pada PC1 mencapai 90,7%, sedangkan kelas *true* hanya sebanyak 9,3%. Dari kelima versi *ranking* fitur pada PC1, jika diambil top 10 fitur, maka metode *GR* dan *SU* tedapat kesamaan anggota fitur paling banyak. Rincian urutan *ranking* fitur tertinggi hingga terendah pada PC1 dapat dilihat pada Tabel 4.10.

Tabel 4.10 *Ranking* Fitur Pada PC1

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
1	0,0639	14	0,0753	16	90,846	14	0,2477	16	0,0896	14
2	0,0495	3	0,0705	14	90,846	15	0,2067	18	0,066	16
3	0,0448	15	0,064	9	90,328	3	0,1951	9	0,0649	9
4	0,0426	9	0,0613	18	90,328	4	0,1603	21	0,0635	15
5	0,0416	16	0,0576	1	90,328	2	0,1458	1	0,0616	1
6	0,0411	1	0,0513	15	89,983	7	0,1437	14	0,0542	18
7	0,0342	18	0,0485	13	89,983	6	0,1359	15	0,0508	13
8	0,0337	13	0,0435	19	89,983	21	0,1239	2	0,0443	19
9	0,0291	19	0,0417	6	89,983	5	0,1156	13	0,0438	3
10	0,0284	6	0,0397	11	89,983	9	0,1068	8	0,043	6
11	0,0276	17	0,0377	17	89,983	8	0,0911	7	0,0411	17
12	0,0215	11	0,0359	5	89,983	11	0,0668	4	0,0344	11
13	0,0201	10	0,0322	20	89,983	10	0,0615	17	0,0318	5
14	0,0201	12	0,032	10	89,983	17	0,0601	11	0,0311	12
15	0,0201	4	0,032	12	89,983	19	0,0587	10	0,0311	10
16	0,0201	5	0,0281	4	89,983	18	0,0587	12	0,0301	4
17	0,0176	20	0,0237	2	89,983	16	0,0537	20	0,028	20
18	0,0175	2	0,0184	3	89,983	20	0,0421	5	0,026	2
19	0,013	8	0,0178	8	89,983	13	0,0338	6	0,0193	8
20	0,0129	7	0,0177	21	89,983	12	0,0323	19	0,0179	7
21	0,0107	21	0,0135	7	89,983	1	0,0132	3	0,0166	21

E. Pemeringkatan Fitur Pada PC2

Pada PC2 terdapat 862 data dan 36 fitur. Adapun persentase kelas *false* pada PC2 mencapai 97,4%, sedangkan kelas *true* hanya sebanyak 2,6%. Dari kelima versi *ranking* fitur pada PC2, jika diambil top 10 fitur, maka metode *GR* dan *SU* tedapat kesamaan anggota fitur paling banyak. Rincian urutan *ranking* fitur tertinggi hingga terendah pada PC2 dapat dilihat pada Tabel 4.11.

Tabel 4.11 *Ranking* Fitur Pada PC2

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
1	0,0236986	10	0,0229116	35	97,4478	18	0,41183	35	0,0337472	10
2	0,0191432	15	0,0208507	4	97,4478	17	0,33817	25	0,0268966	35
3	0,0149528	35	0,0198651	30	97,4478	16	0,30831	9	0,0246799	15
4	0,0148652	16	0,0190239	31	97,4478	35	0,29301	29	0,0244096	4
5	0,0135659	4	0,0187909	32	97,4478	22	0,28873	7	0,0234553	23
6	0,0133893	23	0,0182223	34	97,4478	19	0,28246	4	0,0234553	19
7	0,0133893	19	0,017034	3	97,4478	14	0,26562	17	0,0228542	3
8	0,0129118	3	0,0167814	36	97,4478	13	0,25432	22	0,0227327	34
9	0,0127478	36	0,0167078	24	97,4478	5	0,25415	12	0,0225582	36
10	0,0127271	34	0,0161981	23	97,4478	11	0,22068	18	0,0222028	31
11	0,0123353	31	0,0161981	19	97,4478	8	0,20431	32	0,0208274	16
12	0,0116017	17	0,0161106	33	97,4478	9	0,19421	30	0,0204656	17
13	0,0114225	24	0,0157328	21	97,4478	10	0,17173	31	0,0204567	32
14	0,0114159	33	0,0156521	20	97,4478	20	0,16459	21	0,0204487	24
15	0,0112809	32	0,014866	17	97,4478	21	0,14668	2	0,0203589	33
16	0,0106717	30	0,0134677	12	97,4478	7	0,14195	27	0,0195437	30
17	0,0106491	11	0,0131821	18	97,4478	31	0,14195	5	0,0186863	20
18	0,0104081	20	0,012829	25	97,4478	30	0,14081	11	0,0182815	12
19	0,0103442	12	0,0123709	28	97,4478	32	0,1289	8	0,0182574	11
20	0,0090656	28	0,0119131	1	97,4478	23	0,12169	24	0,0162582	21
21	0,008919	21	0,0117742	7	97,4478	4	0,11922	33	0,0161079	28
22	0,0086233	1	0,0113031	6	97,4478	29	0,11454	34	0,0154128	18
23	0,0085646	18	0,01096	11	97,4478	28	0,11095	13	0,0153429	1
24	0,0081332	25	0,0100431	10	97,4478	27	0,10515	23	0,0146721	25
25	0,0080472	6	0,0095277	22	97,4478	26	0,10515	19	0,0143439	6
26	0,0077892	7	0,0083475	29	97,3318	24	0,10423	20	0,0139918	7
27	0,0065777	13	0,007691	26	97,3318	15	0,10271	26	0,0109613	13
28	0,0057282	22	0,0072459	5	97,3318	25	0,09758	3	0,0103861	22
29	0,005567	26	0,0072459	27	97,2158	3	0,09495	36	0,009905	26
30	0,0048461	27	0,0063978	2	97,2158	6	0,09373	6	0,0086954	27
31	0,0048461	5	0,0060737	9	97,2158	36	0,08923	1	0,0086954	5
32	0,0045668	29	0,0059608	16	97,2158	1	0,08406	28	0,0083505	29
33	0,0045462	2	0,0059518	15	97,2158	33	0,0619	14	0,0081051	2
34	0,0042364	9	0,0056328	13	97,0998	2	0,01963	15	0,007568	9
35	0,0038488	8	0,0056312	8	97,0998	12	0,01925	10	0,0068903	8
36	0,0000362	14	0,000017	14	97,0998	34	0,00978	16	0,000053	14

F. Pemeringkatan Fitur Pada PC3

Pada PC3 terdapat 1281 data dan 37 fitur. Adapun persentase kelas *false* pada PC3 mencapai 88,2%, sedangkan kelas *true* hanya sebanyak 11,8%. Dari kelima versi *ranking* fitur pada PC3, jika diambil top 10 fitur, maka metode *GR*

dan *SU* tedapat kesamaan anggota fitur paling banyak. Rincian urutan *ranking* fitur tertinggi hingga terendah pada PC3 dapat dilihat pada Tabel 4.12.

Tabel 4.12 *Ranking* Fitur Pada PC3

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
1	0,04678	1	0,09058	1	88,603	36	0,208	1	0,07365	1
2	0,04319	4	0,07297	33	88,368	10	0,1869	26	0,06244	4
3	0,03903	33	0,07289	18	88,212	14	0,1826	10	0,06099	33
4	0,03853	18	0,06794	35	88,212	13	0,1689	18	0,06036	18
5	0,03645	36	0,06035	25	88,212	12	0,1629	30	0,05571	35
6	0,03546	35	0,05896	4	88,212	15	0,1604	36	0,05288	36
7	0,0314	5	0,05645	31	88,212	16	0,1525	9	0,04906	25
8	0,03115	25	0,05625	22	88,212	17	0,1499	5	0,04757	5
9	0,02918	31	0,05614	21	88,212	11	0,1436	11	0,04594	31
10	0,02873	22	0,05334	32	88,212	37	0,1388	35	0,04534	22
11	0,02852	21	0,05132	5	88,212	18	0,1326	12	0,04506	21
12	0,02731	32	0,05036	36	88,212	4	0,121	4	0,04307	32
13	0,02532	37	0,04976	37	88,212	2	0,1196	3	0,04	37
14	0,02357	16	0,04227	24	88,212	3	0,112	8	0,03561	16
15	0,02138	20	0,04227	20	88,212	5	0,1017	17	0,03382	20
16	0,02138	24	0,03809	16	88,212	8	0,0887	23	0,03382	24
17	0,01943	34	0,03314	13	88,212	6	0,0844	24	0,02938	34
18	0,01815	19	0,03154	34	88,212	7	0,0844	20	0,02719	13
19	0,01738	29	0,03149	29	88,212	9	0,0843	16	0,02697	29
20	0,01731	13	0,02869	30	88,212	19	0,082	14	0,02581	19
21	0,01624	11	0,02723	11	88,212	20	0,0798	19	0,02475	11
22	0,01623	26	0,025	2	88,212	29	0,0761	29	0,02448	30
23	0,01575	30	0,02458	26	88,212	31	0,0758	33	0,02413	26
24	0,01573	3	0,02413	7	88,212	32	0,0743	37	0,02344	7
25	0,01572	23	0,02368	3	88,212	33	0,0723	15	0,02335	3
26	0,01571	7	0,02336	19	88,212	34	0,0706	21	0,02243	23
27	0,01351	2	0,02188	28	88,212	35	0,0632	22	0,02106	2
28	0,01151	28	0,02093	27	88,212	30	0,0579	31	0,01805	28
29	0,01105	27	0,02084	6	88,212	28	0,0532	27	0,01732	27
30	0,01097	6	0,02047	23	88,212	21	0,0503	13	0,0172	6
31	0,01024	8	0,01694	9	88,212	27	0,0492	7	0,0155	8
32	0,00937	9	0,01667	8	88,212	22	0,0449	32	0,01454	9
33	0,00822	10	0,01199	10	88,212	23	0,0436	25	0,0121	10
34	0,0058	12	0,00988	17	88,212	24	0,0435	2	0,00875	12
35	0,00548	17	0,0093	12	88,212	25	0,0366	28	0,0085	17
36	0,0041	14	0,00539	14	88,212	26	0,0351	6	0,00587	14
37	0,00401	15	0,00482	15	88,212	1	0,0338	34	0,00558	15

G. Pemeringkatan Fitur Pada PC4

Pada PC4 terdapat 1228 data dan 37 fitur. Adapun persentase kelas *false* pada PC4 mencapai 88,6%, sedangkan kelas *true* hanya sebanyak 13,4%. Dari kelima versi *ranking* fitur pada PC4, jika diambil top 10 fitur, maka metode *GR* dan *SU* tedapat kesamaan anggota fitur paling banyak. Rincian urutan *ranking* fitur tertinggi hingga terendah pada PC4 dapat dilihat pada Tabel 4.13.

Tabel 4.13 *Ranking* Fitur Pada PC4

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
1	0,15155	4	0,18613	4	88,518	4	0,48244	4	0,20736	4

Rank	GR		IG		OR		RFF		SU	
	Score	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur	*Score*	Fitur
2	0,07168	36	0,11491	36	86,971	37	0,34443	36	0,1059	36
3	0,06495	10	0,07918	1	86,889	20	0,2495	8	0,08463	10
4	0,04989	6	0,07669	8	86,889	21	0,23477	30	0,07207	6
5	0,04989	28	0,07359	28	86,889	31	0,21825	1	0,07207	28
6	0,04914	27	0,07359	6	86,889	22	0,20001	27	0,07095	9
7	0,04907	9	0,07261	9	86,889	25	0,19624	10	0,07071	27
8	0,04591	5	0,07148	27	86,889	24	0,1961	9	0,06623	1
9	0,04341	1	0,06885	10	86,808	16	0,19576	6	0,06155	5
10	0,04337	37	0,06283	30	86,645	8	0,19576	28	0,06129	8
11	0,03963	8	0,05646	37	86,645	3	0,12851	5	0,06043	37
12	0,03424	30	0,05294	5	86,645	12	0,1271	35	0,05231	30
13	0,02819	25	0,04897	35	86,645	13	0,1063	37	0,0408	35
14	0,0279	22	0,03763	25	86,645	14	0,10162	11	0,03957	25
15	0,02711	21	0,03717	18	86,645	2	0,09388	19	0,03899	22
16	0,02671	35	0,03696	32	86,645	5	0,09374	23	0,03748	21
17	0,02607	16	0,0367	22	86,645	9	0,09163	20	0,03612	18
18	0,02514	20	0,03444	21	86,645	11	0,09163	24	0,03579	16
19	0,02514	24	0,03387	24	86,645	15	0,08972	32	0,03539	24
20	0,02492	18	0,03387	20	86,645	7	0,08391	33	0,03539	20
21	0,02172	31	0,03353	31	86,645	6	0,08289	18	0,0322	32
22	0,02138	32	0,03237	16	86,645	10	0,08198	34	0,03177	31
23	0,02133	34	0,03119	34	86,645	19	0,08072	29	0,03074	34
24	0,01712	23	0,03042	33	86,645	17	0,08039	22	0,0256	33
25	0,01681	33	0,02556	19	86,645	33	0,07656	25	0,02337	23
26	0,01627	12	0,02545	26	86,645	32	0,07187	14	0,0226	26
27	0,01583	7	0,02088	23	86,645	34	0,0683	17	0,02211	19
28	0,0151	26	0,01903	17	86,645	18	0,0668	7	0,0203	12
29	0,01465	19	0,01775	13	86,645	35	0,06604	2	0,01972	7
30	0,01409	11	0,01698	11	86,645	30	0,06356	13	0,01916	11
31	0,01404	2	0,01531	12	86,645	29	0,0593	26	0,01762	17
32	0,01195	17	0,01486	29	86,645	28	0,05471	16	0,01722	2
33	0,01135	13	0,01483	7	86,645	27	0,05437	21	0,01666	13
34	0,00947	29	0,01263	2	86,645	36	0,05399	31	0,01391	29
35	0,00936	14	0,00933	3	86,645	23	0,0328	3	0,01187	14
36	0,00826	3	0,0092	14	86,645	26	0,02734	12	0,011	3
37	0,00497	15	0,00191	15	86,645	1	0,00878	15	0,00402	15

4.2. Perancangan Uji Coba

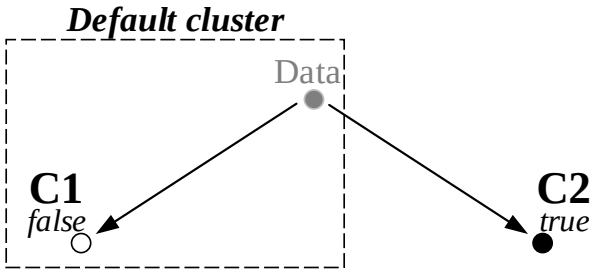
Setelah semua fitur diurutkan berdasarkan pemeringkatan lima metode seleksi fitur, selanjutnya ditentukan skenario pengujian untuk mengetahui pengaruh usulan kombinasi metode terhadap nilai *balance*. Berdasarkan Gambar 3.17 pada BAB 3, skenario pengujian pada penelitian ini dilakukan hanya pada tahap klasifikasi saja (tidak termasuk tahap pra-proses). Hal tersebut dilakukan untuk mengetahui berapa jumlah fitur terbaik (paling relevan) yang dapat meningkatkan nilai *balance* di proses klasifikasi menggunakan *CBC*.

Adapun jumlah *fold* untuk melakukan *cross validation* pada pengujian ini adalah 5 *folds*. Pada tahap ini, *cross validation* dilakukan dengan menggunakan

bantuan fungsi *crossvalind* di MATLAB. Fungsi tersebut dapat digunakan untuk membagi data menjadi data *training* dan data *testing* berdasarkan persebaran label kelas di setiap *fold*.

Pada dasarnya proses *CBC* adalah proses *clustering*. Metode *clustering* yang digunakan pada penelitian ini adalah *k-means*. Untuk mendapatkan hasil *k-means*, maka diperlukan iterasi pencarian titik *cluster center* baru hingga konvergen, yaitu kondisi di mana tidak ada perubahan anggota data di titik *cluster center* baru. Namun, jika konvergensi hanya diperiksa dari nilai iterasi -1, maka ditemukan kondisi dimana *CBC* mencapai satu juta iterasi dengan anggota *cluster* yang masih belum konvergen. Hal ini disebabkan oleh proses pembulatan nilai *mean* yang sudah dijelaskan pada Sub-bab 3.2.6. Untuk itu, pada penelitian ini digunakan iterasi -5 sebagai *threshold* pencarian konvergensi. Jadi, jika iterasi -1 tidak kunjung konvergen, maka iterasi yang sekarang akan dibandingkan dengan iterasi -5 sebelumnya secara iteratif. Ketika iterasi belum mencapai -5 dan sudah konvergen, maka iterasi dihentikan.

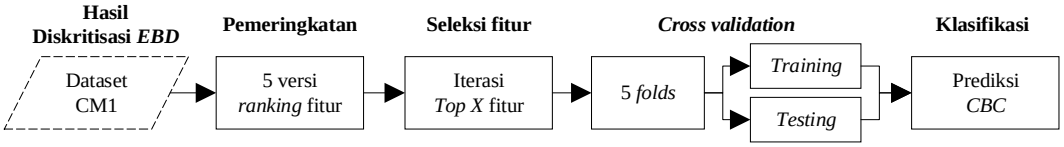
Adapun untuk menentukan keanggotaan data terhadap suatu *cluster* tertentu, maka digunakan perhitungan jarak antara data terhadap titik *cluster*. Jarak data yang paling terdekat dengan *cluster* tertentu, akan dikelompokkan ke dalam anggota *cluster* itu. Perhitungan jarak yang digunakan pada penelitian ini adalah *hamming distance*. Sedangkan pada penelitian *CBC* sebelumnya menggunakan *euclidean distance* (Singh & Verma 2014). Adapun jumlah titik *cluster* pada penelitian ini ada dua, yaitu C1 dan C2. Diasumsikan bahwa titik *cluster* C1 mewakili kelompok data dengan kelas *false*, sedangkan *cluster* C2 mewakili kelompok data dengan kelas *true*. Penentuan keanggotaan data terhadap *cluster* tertentu, dinyatakan dengan kondisi perhitungan jarak C1 > C2. Jika jarak data terhadap C1 lebih jauh dari C2, maka data itu termasuk ke dalam anggota C2 dan sebaliknya. Namun, ketika terdapat kesamaan jarak antara data terhadap masing-masing titik *cluster*, secara *default* data tersebut akan masuk ke dalam anggota C1. Artinya, sistem prediksi menilai keanggotaan data yang abu-abu (jarak data ke C1 dan C2 sama) adalah sebagai kelas *false*. Ilustrasi penentuan anggota *cluster* yang abu-abu dapat dilihat pada Gambar 4.2.



Gambar 4.2 Ilustrasi Penentuan Keanggotaan Data yang Abu-abu

4.2.1. Skenario Pengujian di CM1

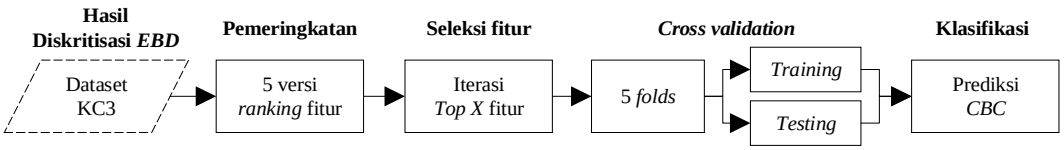
Skenario pengujian yang pertama menggunakan dataset CM1. Skenario ini dilakukan dengan mengkombinasikan metode *EBD* dua fase dan kelima metode seleksi fitur *GR*, *IG*, *OR*, *RFF*, dan *SU*. Setelah itu, fitur diseleksi sejumlah *Top X* fitur secara iteratif untuk dijadikan sebagai *input* model klasifikasi *CBC*. Sejumlah 5 *folds* digunakan untuk melakukan *cross validation*. Adapun *default* prediksi pada pengujian ini adalah C1 yaitu kelas *false*. Ilustrasi skenario pengujian dengan menggunakan dataset CM1 di Gambar 4.3.



Gambar 4.3 Tahapan Skenario Pengujian di CM1

4.2.2. Skenario Pengujian di KC3

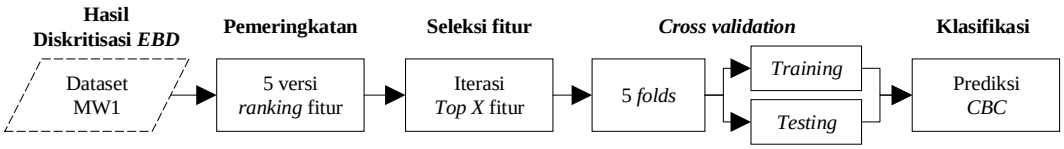
Skenario pengujian yang kedua menggunakan dataset KC3. Skenario ini dilakukan dengan mengkombinasikan metode *EBD* dua fase dan kelima metode seleksi fitur *GR*, *IG*, *OR*, *RFF*, dan *SU*. Setelah itu, fitur diseleksi sejumlah *Top X* fitur secara iteratif untuk dijadikan sebagai *input* model klasifikasi *CBC*. Sejumlah 5 *folds* digunakan untuk melakukan *cross validation*. Adapun *default* prediksi pada pengujian ini adalah C1 yaitu kelas *false*. Ilustrasi skenario pengujian dengan menggunakan dataset KC3 di Gambar 4.4.



Gambar 4.4 Tahapan Skenario Pengujian di KC3

4.2.3. Skenario Pengujian di MW1

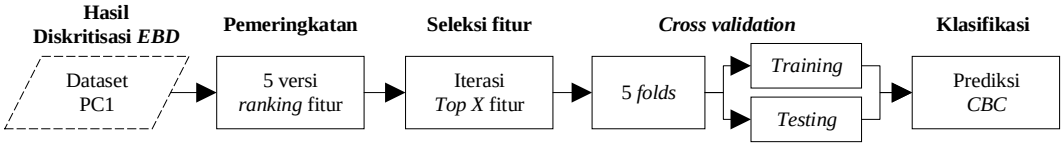
Skenario pengujian yang ketiga menggunakan dataset MW1. Skenario ini dilakukan dengan mengkombinasikan metode *EBD* dua fase dan kelima metode seleksi fitur *GR*, *IG*, *OR*, *RFF*, dan *SU*. Setelah itu, fitur diseleksi sejumlah *Top X* fitur secara iteratif untuk dijadikan sebagai *input* model klasifikasi *CBC*. Sejumlah 5 *folds* digunakan untuk melakukan *cross validation*. Adapun *default* prediksi pada pengujian ini adalah MW1 yaitu kelas *false*. Ilustrasi skenario pengujian dengan menggunakan dataset KC3 di Gambar 4.5.



Gambar 4.5 Tahapan Skenario Pengujian di MW1

4.2.4. Skenario Pengujian di PC1

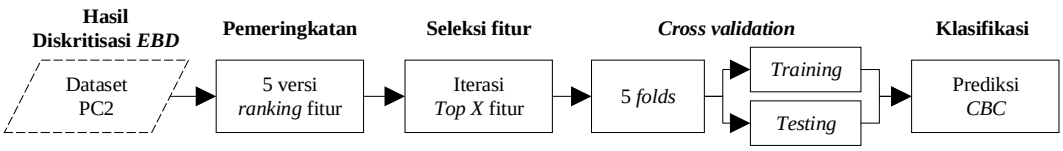
Skenario pengujian yang keempat menggunakan dataset PC1. Skenario ini dilakukan dengan mengkombinasikan metode *EBD* dua fase dan kelima metode seleksi fitur *GR*, *IG*, *OR*, *RFF*, dan *SU*. Setelah itu, fitur diseleksi sejumlah *Top X* fitur secara iteratif untuk dijadikan sebagai *input* model klasifikasi *CBC*. Sejumlah 5 *folds* digunakan untuk melakukan *cross validation*. Adapun *default* prediksi pada pengujian ini adalah C1 yaitu kelas *false*. Ilustrasi skenario pengujian dengan menggunakan dataset PC1 di Gambar 4.6.



Gambar 4.6 Tahapan Skenario Pengujian di PC1

4.2.5. Skenario Pengujian di PC2

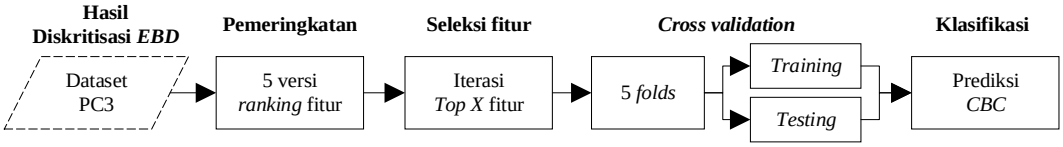
Skenario pengujian yang kelima menggunakan dataset PC2. Skenario ini dilakukan dengan mengkombinasikan metode *EBD* dua fase dan kelima metode seleksi fitur *GR*, *IG*, *OR*, *RFF*, dan *SU*. Setelah itu, fitur diseleksi sejumlah *Top X* fitur secara iteratif untuk dijadikan sebagai *input* model klasifikasi *CBC*. Sejumlah 5 *folds* digunakan untuk melakukan *cross validation*. Adapun *default* prediksi pada pengujian ini adalah C1 yaitu kelas *false*. Ilustrasi skenario pengujian dengan menggunakan dataset PC2 di Gambar 4.7.



Gambar 4.7 Tahapan Skenario Pengujian di PC2

4.2.6. Skenario Pengujian di PC3

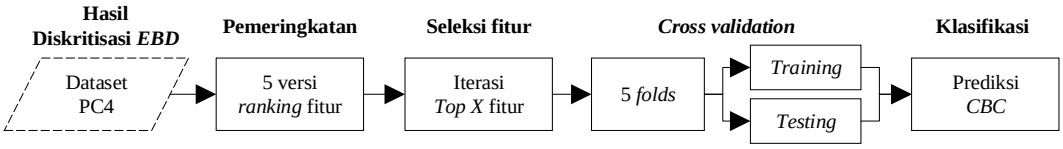
Skenario pengujian yang keenam menggunakan dataset PC3. Skenario ini dilakukan dengan mengkombinasikan metode *EBD* dua fase dan kelima metode seleksi fitur *GR*, *IG*, *OR*, *RFF*, dan *SU*. Setelah itu, fitur diseleksi sejumlah *Top X* fitur secara iteratif untuk dijadikan sebagai *input* model klasifikasi *CBC*. Sejumlah 5 *folds* digunakan untuk melakukan *cross validation*. Adapun *default* prediksi pada pengujian ini adalah C1 yaitu kelas *false*. Ilustrasi skenario pengujian dengan menggunakan dataset PC3 di Gambar 4.8.



Gambar 4.8 Tahapan Skenario Pengujian di PC3

4.2.7. Skenario Pengujian di PC4

Skenario pengujian yang ketujuh menggunakan dataset PC3. Skenario ini dilakukan dengan mengkombinasikan metode *EBD* dua fase dan kelima metode seleksi fitur *GR*, *IG*, *OR*, *RFF*, dan *SU*. Setelah itu, fitur diseleksi sejumlah *Top X* fitur secara iteratif untuk dijadikan sebagai *input* model klasifikasi *CBC*. Sejumlah 5 *folds* digunakan untuk melakukan *cross validation*. Adapun *default* prediksi pada pengujian ini adalah C1 yaitu kelas *false*. Ilustrasi skenario pengujian dengan menggunakan dataset PC3 di Gambar 4.9.



Gambar 4.9 Tahapan Skenario Pengujian di PC4

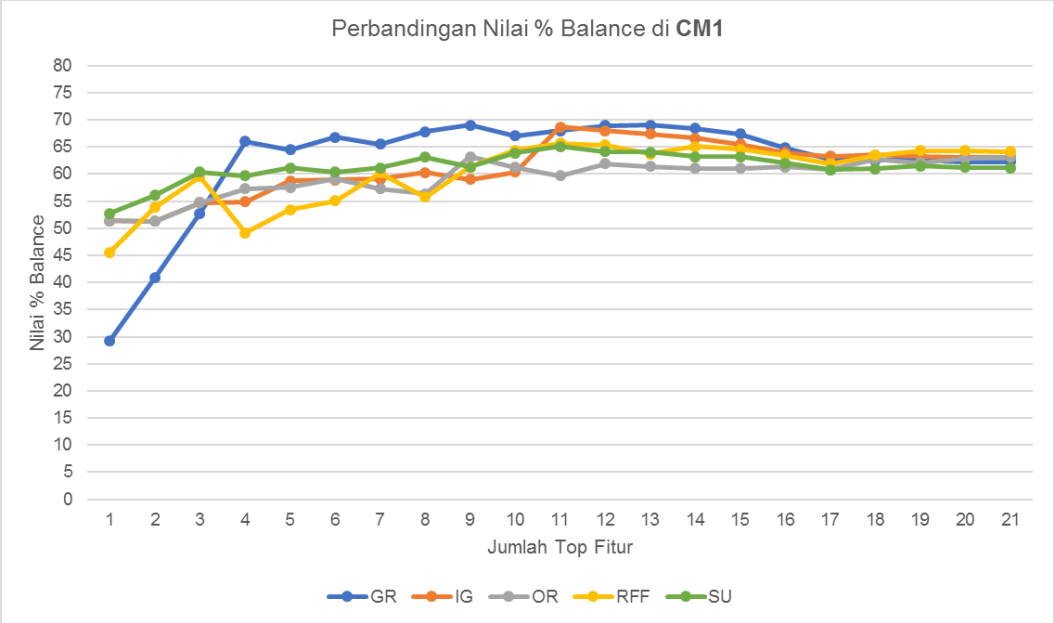
4.3. Analisis Hasil

Pada bagian ini dijelaskan tentang hasil dari setiap skenario pengujian. Setiap pengujian akan dicari *Top X* fitur *(trend)* dengan *balance* tertinggi secara

iteratif dari setiap metode seleksi fitur. Serta akan dilakukan perbandingan nilai *balance* terbaik antar metode seleksi fitur. Hal tersebut dilakukan untuk dapat menjawab rumusan masalah pada penelitian ini. Lalu, jika dimensi dataset yang digunakan pada skenario sama dengan penelitian sebelumnya, maka hasil *balance* terbaik di skenario ini akan dibandingkan dengan hasil *balance* penelitian sebelumnya, yaitu kombinasi *EBD* dan *CBC* tanpa seleksi fitur.

4.3.1. Hasil Skenario Pengujian di CM1

- *Balance* CM1



Gambar 4.10 Nilai *Balance* Pada Skenario Pengujian di CM1

Berdasarkan Gambar 4.10, *trend* nilai *balance* di semua seleksi fitur meningkat di *top* 9 fitur hingga *top* 11 fitur. Selain itu, *trend* menunjukkan bahwa nilai *balance* metode seleksi fitur *GR* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *balance* tertinggi dari setiap seleksi fitur yaitu *GR* 9 fitur menghasilkan *balance* 69,1%, *IG* 11 fitur menghasilkan *balance* 68,7%, *OR* 9 fitur menghasilkan *balance* 63,2%, *RFF* 11 fitur menghasilkan *balance* 65,7%, dan *SU* 11 fitur menghasilkan *balance* 65%. Adapun detil nilai *balance*, disajikan pada Tabel 4.14.

Tabel 4.14 Nilai *Balance* Pada CM1

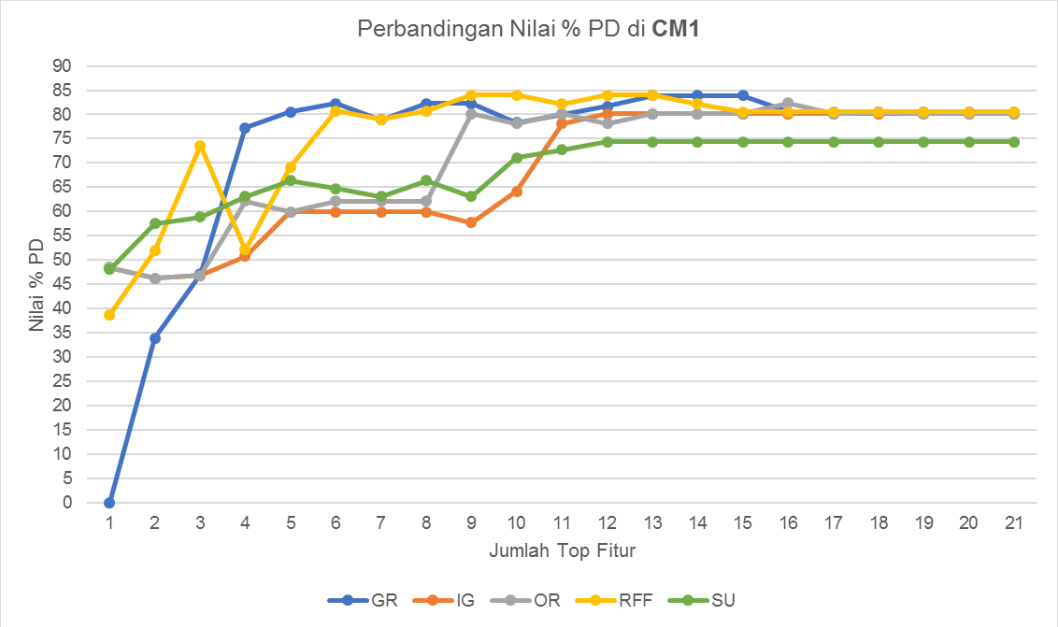
Balance (CM1)					
Top Fitur	**GR**	**IG**	**OR**	**RFF**	**SU**
1	29,28932	51,39914	51,39914	45,55259	52,78102
2	40,82879	51,29303	51,29303	53,85194	56,09978
3	52,78369	54,74702	54,74702	59,54179	60,32849
4	66,01331	54,92766	57,28417	49,18319	59,70662
5	64,49894	58,74671	57,51805	53,40581	61,17074
6	66,82336	58,90653	59,0899	55,07658	60,36651
7	65,47776	59,12549	57,2567	60,1658	61,11568
8	67,80748	60,26882	56,33941	55,75394	63,0871
9	**69,0569**	59,00005	**63,1951**	61,30792	61,32832
10	67,02389	60,36646	61,24865	64,41385	63,88322
11	68,01684	**68,7041**	59,67521	**65,7015**	**65,0469**
12	68,9596	68,00332	61,88187	65,37529	64,12744
13	69,01054	67,37678	61,39403	63,70492	64,05942
14	68,44842	66,59421	61,05038	65,04948	63,20095
15	67,41436	65,54478	61,05038	64,55803	63,20095
16	64,80936	63,99	61,30692	63,43998	62,00794
17	62,75511	63,32326	60,79307	61,95388	60,79349
18	62,51001	63,55859	62,79862	63,43998	60,95671
19	62,51001	63,32326	62,08318	64,3251	61,48042
20	62,29523	63,08766	62,87283	64,3251	61,24872
21	62,29523	63,08766	63,08766	64,11868	61,1041

Berdasarkan Tabel 4.14, fitur terbaik CM1 berada pada Top 9 hingga 11 fitur. Jumlah *top* fitur pada *GR* dan *OR* sama, yaitu 9. Namun, *IG*, *RFF*, dan *SU* dapat menghasilkan nilai *balance* terbaik di *top* 11 fitur. Nilai *balance* yang berwarna kuning artinya nilai *balance* tertinggi di metode seleksi fitur tertentu. Sedangkan nilai *balance* yang berwarna biru artinya nilai *balance* tertinggi di antara empat metode seleksi fitur lainnya. Pada Tabel 4.14 ditunjukkan bahwa hanya *Top* 9 fitur pada CM1 yang relevan dengan informasi untuk prediksi kesalahan perangkat lunak menggunakan *CBC,* yaitu *Top* fitur *GR*. Berdasarkan peringkat fitur *GR* di Tabel 4.14, maka 9 fitur tersebut disajikan pada Tabel 4.15.

Tabel 4.15 Fitur Terbaik Pada CM1

Peringkat	**No Fitur**	**Nama Fitur**
1	16	C&SLOC
2	19	N1
3	11	B
4	14	CLOC
5	6	V
6	18	n2
7	17	n1
8	9	I
9	1	LOC

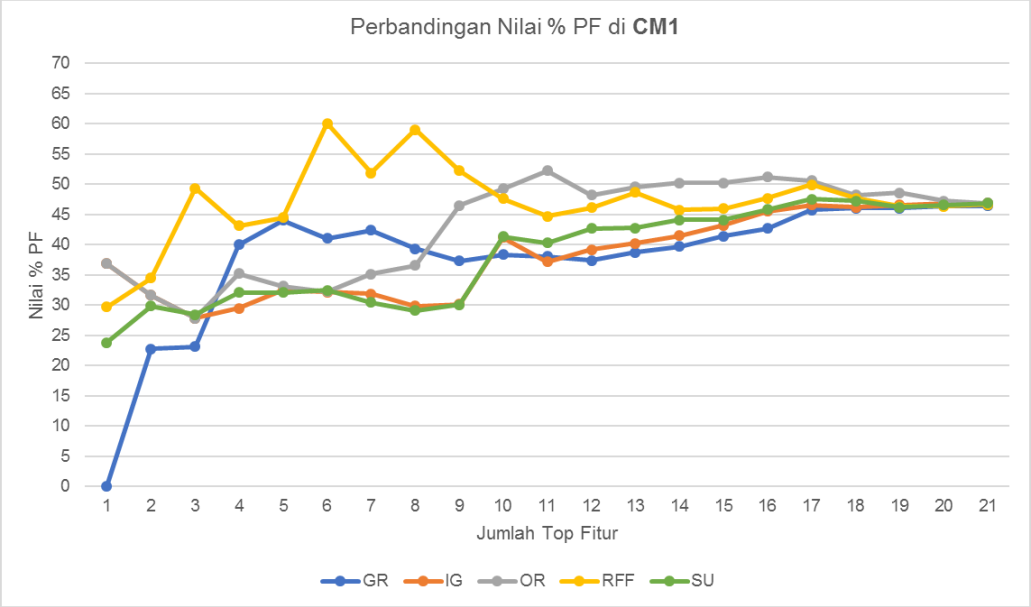
- *Pd* CM1



Gambar 4.11 Nilai *Pd* Pada Skenario Pengujian di CM1

Berdasarkan Gambar 4.11, *trend* nilai *pd* di semua seleksi fitur meningkat di *top* 12 fitur hingga *top* 16 fitur. Selain itu, *trend* menunjukkan bahwa nilai *pd* metode seleksi fitur *RFF* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *pd* tertinggi dari setiap seleksi fitur yaitu *GR* 13 fitur menghasilkan *pd* 83,89%, *IG* 12 fitur menghasilkan *pd* 68,7%, *OR* 9 fitur menghasilkan *pd* 63,2%, *RFF* 11 fitur menghasilkan *pd* 65,7%, dan *SU* 11 fitur menghasilkan *pd* 65%.

- *Pf* CM1

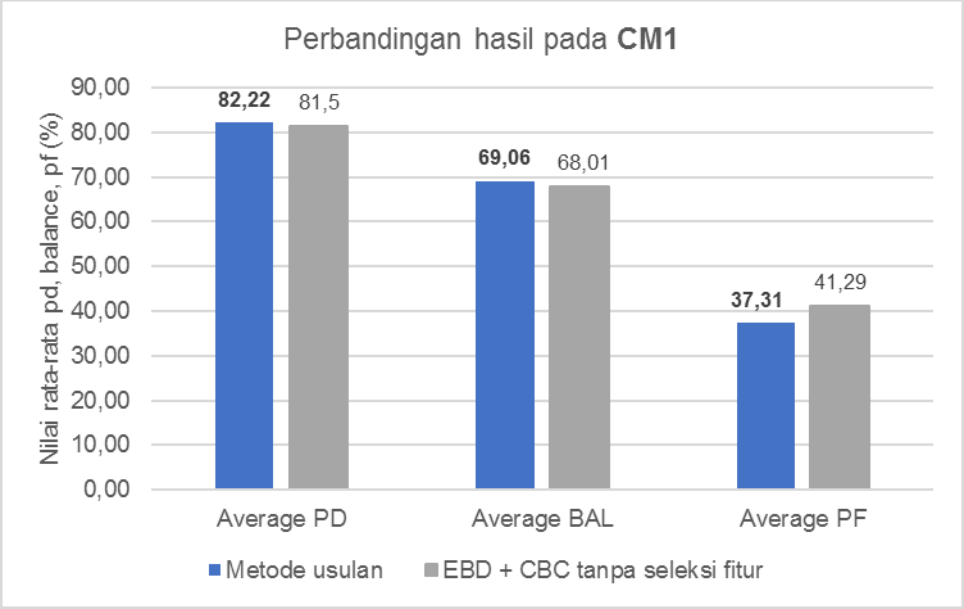


Gambar 4.12 Nilai *Pf* Pada Skenario Pengujian di CM1

Berdasarkan Gambar 4.12, nilai *pf* dari semua metode seleksi fitur cenderung menurun ketika *top* fitur dikurangi. Hanya metode seleksi fitur *RFF* yang menghasilkan nilai *pf* tertinggi yaitu pada *top* 6 fitur 60%.

- Perbandingan Hasil Penelitian di CM1

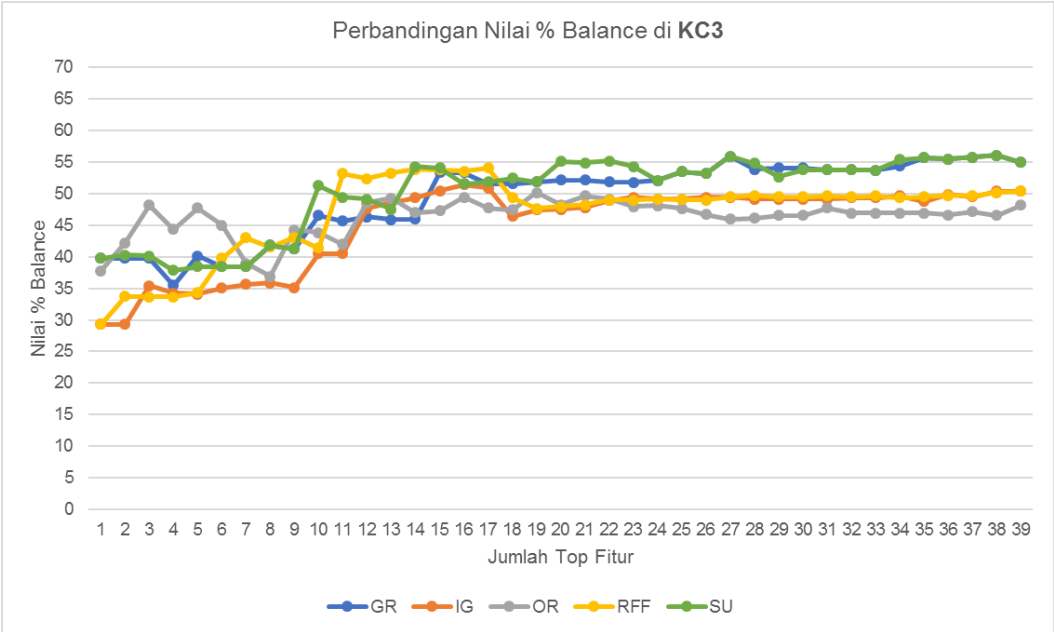
Jika dibandingkan dengan penelitian sebelumnya (Singh & Verma 2014) pada dataset CM1 yang sama, maka penelitian ini dapat menghasilkan nilai *balance* yang lebih baik, yaitu dengan peningkatan *balance* sebesar 1,05%. Ada pun perbandingan nilai *balance* dapat dilihat di Gambar 4.13.



Gambar 4.13 Perbandingan Nilai *Pd*, *Balance*, *Pf* di CM1 Antara Metode Usulan dan *EBD* + *CBC* Tanpa Seleksi Fitur

4.3.2. Hasil Skenario Pengujian di KC3

- *Balance* KC3



Gambar 4.14 Nilai *Balance* Pada Skenario Pengujian di KC3

Berdasarkan Gambar 4.14, *trend* nilai *balance* di semua seleksi fitur meningkat di *top* 16 fitur hingga *top* 38 fitur. Selain itu, *trend* menunjukkan bahwa nilai *balance* metode seleksi fitur *GR* dan *SU* cenderung lebih tinggi dari metode

seleksi fitur yang lain. Adapun rincian nilai *balance* tertinggi dari setiap seleksi fitur yaitu *GR* 38 fitur menghasilkan *balance* 56,07%, *IG* 16 fitur menghasilkan *balance* 51,45%, *OR* 19 fitur menghasilkan *balance* 50,18%, *RFF* 17 fitur menghasilkan *balance* 54,07%, dan *SU* 38 fitur menghasilkan *balance* 56,07%. Adapun detil nilai *balance*, disajikan pada Tabel 4.16.

Tabel 4.16 Nilai *Balance* Pada KC3

Balance (KC3)					
Top Fitur	GR	IG	OR	RFF	SU
1	39,77791	29,28932	37,76463	29,28932	39,77791
2	39,77791	29,28932	42,22529	33,73045	40,28466
3	39,77791	35,40948	48,2261	33,66204	40,14147
4	35,54074	34,22651	44,33107	33,66204	37,87535
5	40,14147	34,1159	47,77596	34,3398	38,44126
6	38,44126	35,08138	45,01689	39,77617	38,44126
7	38,44126	35,66314	38,97346	43,02133	38,44126
8	41,87905	35,85638	36,9229	41,53011	41,87905
9	41,23128	35,13853	44,27006	43,13306	41,23128
10	46,56632	40,55202	43,85731	41,38632	51,27527
11	45,70335	40,55202	41,96693	53,19662	49,4361
12	46,31742	47,6758	48,54992	52,36878	49,17851
13	45,93683	48,59812	49,29443	53,29109	47,52419
14	45,95284	49,34515	46,99452	53,84761	54,33854
15	53,33764	50,47731	47,29195	53,79695	54,07317
16	53,34599	**51,4491**	49,46335	53,54786	51,61312
17	51,61312	50,95523	47,77877	**54,0716**	51,89596
18	51,61312	46,405	47,44441	49,35686	52,47738
19	51,89596	47,43318	**50,182**	47,6457	51,9157
20	52,19686	47,53913	48,23572	47,95313	55,15611
21	52,19686	47,86055	49,65896	48,26423	54,87495
22	51,9157	48,91745	49,18105	49,02673	55,18039
23	51,82832	49,42959	47,96094	49,01487	54,28856
24	52,09169	49,12591	48,10778	49,12576	52,09169
25	53,48782	49,12591	47,65079	49,08041	53,48782
26	53,19343	49,42959	46,73783	48,98144	53,19343
27	55,94735	49,42959	45,97055	49,54701	55,94735
28	53,86185	49,12591	46,11379	49,75702	54,79866
29	54,09586	49,12591	46,58382	49,49513	52,71316
30	54,09586	49,17453	46,58382	49,49513	53,81302
31	53,81302	49,12591	47,78334	49,64357	53,81302
32	53,81302	49,3878	46,90506	49,49513	53,81302
33	53,6939	49,3878	46,90506	49,64357	53,6939
34	54,36269	49,64969	46,90506	49,38918	55,44524
35	55,71207	48,78659	46,90506	49,49513	55,71207
36	55,4797	49,90994	46,63822	49,74953	55,4797
37	55,76831	49,5006	47,1335	49,64805	55,76831
38	**56,0692**	50,42551	46,54769	50,16362	**56,0692**
39	54,98986	50,42551	48,18977	50,42551	54,98986

Berdasarkan Tabel 4.16, fitur terbaik KC3 berada pada *Top* 16 hingga 38 fitur. Jumlah *top* fitur pada *GR* dan *SU* sama, yaitu 38. Namun, *IG*, *OR*, *RFF* memiliki jumlah *Top* fitur yang lebih sedikit yaitu *IG* 16 fitur, *OR* 19 fitur, dan *RFF*

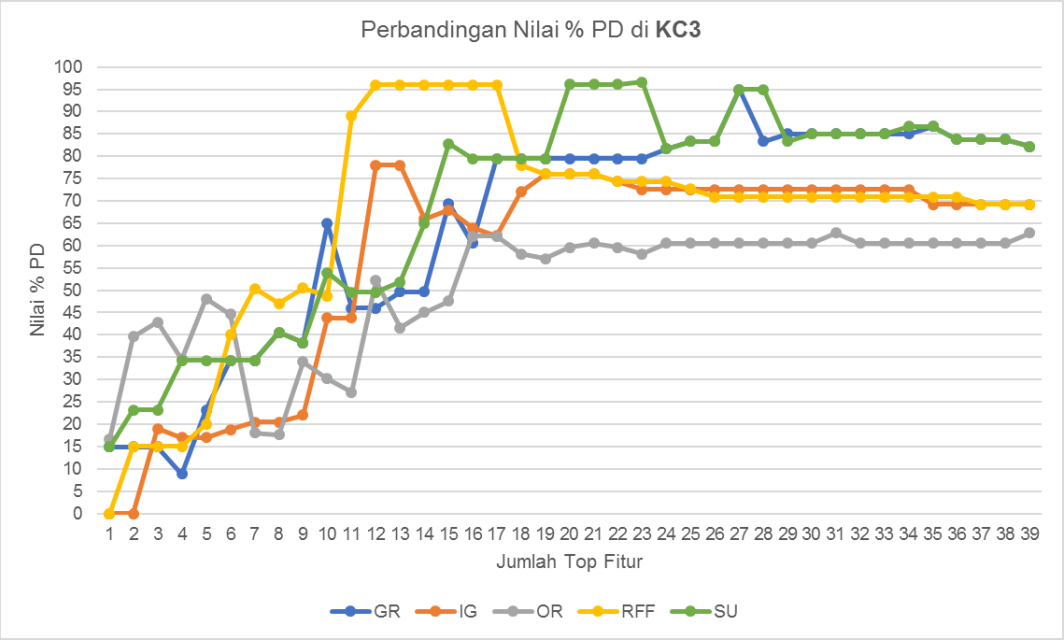
17 fitur. Nilai *balance* yang berwarna kuning artinya nilai *balance* tertinggi di metode seleksi fitur tertentu. Sedangkan nilai *balance* yang berwarna biru artinya nilai *balance* tertinggi di antara metode seleksi fitur lainnya. Pada Tabel 4.16 ditunjukkan bahwa hanya *Top* 38 fitur pada KC3 yang relevan dengan informasi untuk prediksi kesalahan perangkat lunak menggunakan *CBC*, yaitu *Top* fitur *GR* dan *OR*. Berdasarkan peringkat fitur *GR* di Tabel 4.16, maka 38 fitur tersebut disajikan pada Tabel 4.17.

Tabel 4.17 Fitur Terbaik Pada KC3

Peringkat	**No Fitur**	**Nama Fitur**
1	4	C&SLOC
2	16	SLOC
3	39	LOC
4	22	E
5	26	T
6	2	Branch_C
7	24	L
8	27	V
9	7	v(G)
10	14	ev(G)
11	1	BLOC
12	20	I
13	28	Mainsev
14	33	N2
15	34	N1
16	23	B
17	37	nl
18	38	%comment
19	35	n2
20	13	Edge_C
21	31	Node_C
22	3	Call_C
23	18	gdv
24	11	iv(G)
25	21	D
26	36	n1
27	8	vd(G)
28	6	Cond_C
29	30	Mul_Cond_C
30	5	CLOC
31	32	N2
32	9	Dec_C

Peringkat	No Fitur	Nama Fitur
33	29	Mod_Cond_C
34	12	id(G)
35	25	1/D
36	19	gd(G)
37	15	ed(G)
38	10	dd(G)

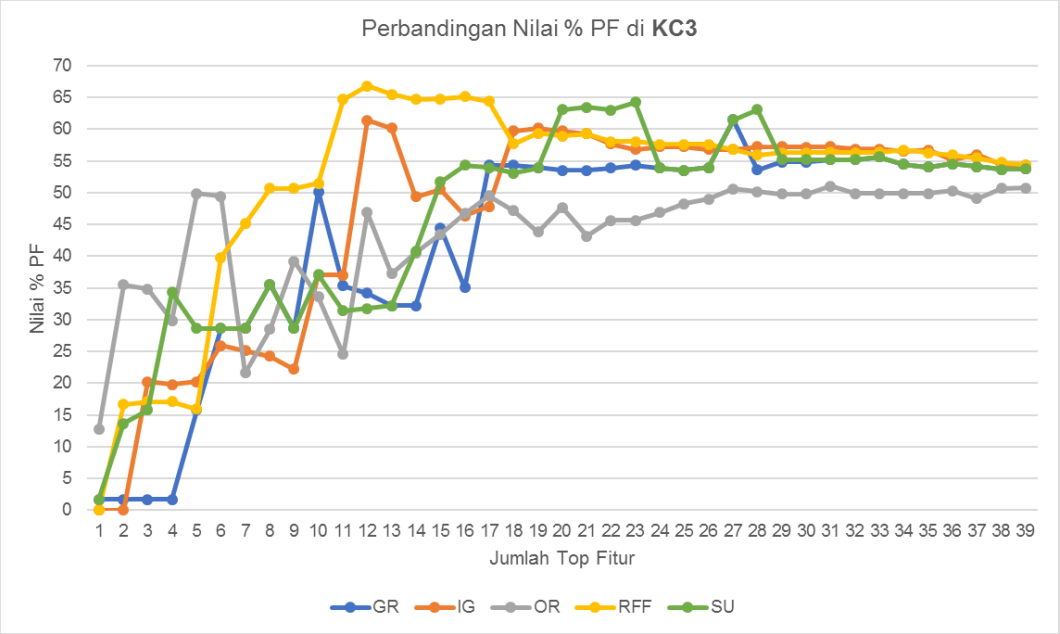
- *Pd* KC3



Gambar 4.15 Nilai *Pd* Pada Skenario Pengujian di KC3

Berdasarkan Gambar 4.15, *trend* nilai *pd* di semua seleksi fitur meningkat di *top* 12 fitur hingga *top* 31 fitur. Selain itu, *trend* menunjukkan bahwa nilai *pd* metode seleksi fitur *SU* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *pd* tertinggi dari setiap seleksi fitur yaitu *GR* 27 fitur menghasilkan *pd* 95%, *IG* 12 fitur menghasilkan *pd* 78%, *OR* 31 fitur menghasilkan *pd* 62,8%, *RFF* 12 fitur menghasilkan *pd* 96%, dan *SU* 23 fitur menghasilkan *pd* 96,67%.

- *Pf* KC3

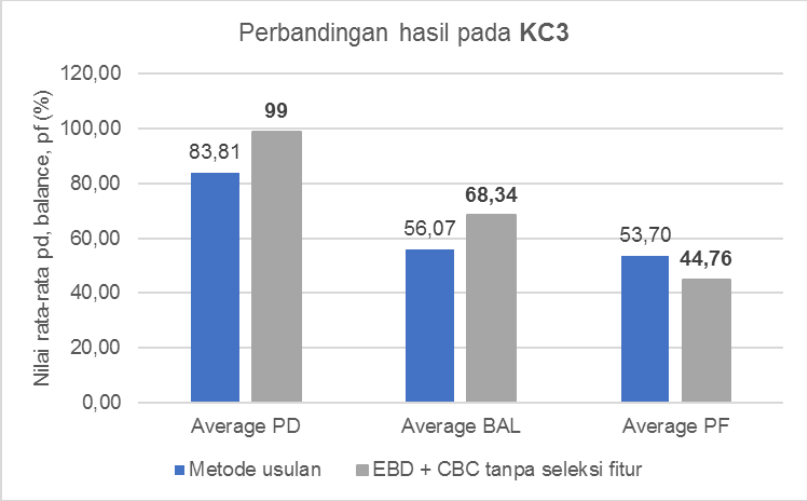


Gambar 4.16 Nilai *Pf* Pada Skenario Pengujian di KC3

Berdasarkan Gambar 4.16, nilai *pf* dari semua metode seleksi fitur cenderung menurun ketika *top* fitur dikurangi. Hanya metode seleksi fitur *RFF* yang menghasilkan nilai *pf* tertinggi yaitu pada *top* 12 fitur menghasilkan 66,81%.

- Perbandingan Hasil Penelitian di KC3

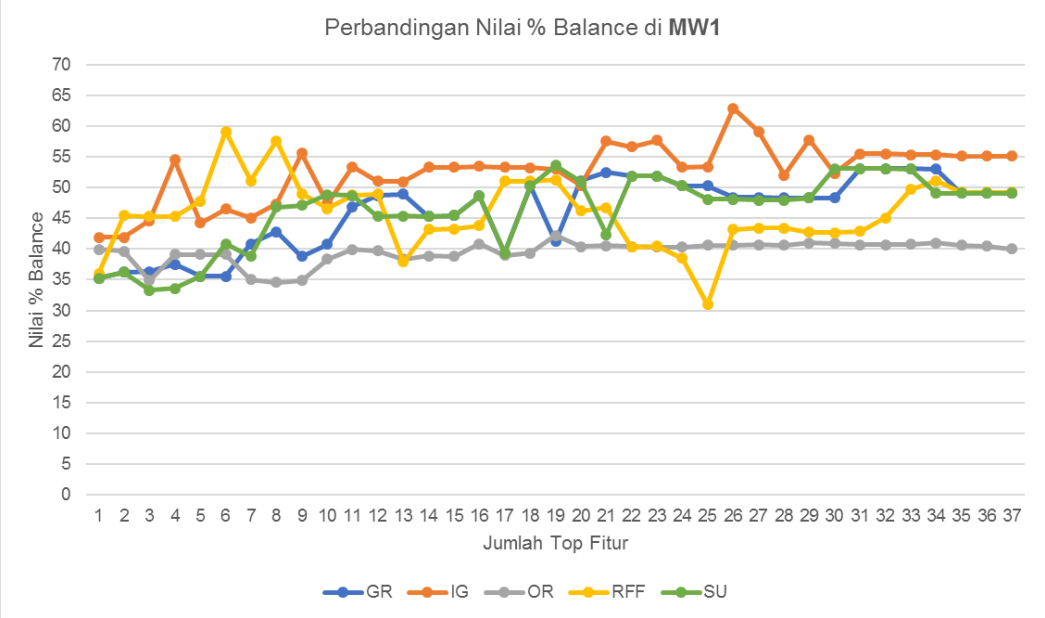
Jika dibandingkan dengan penelitian sebelumnya (Singh & Verma 2014) pada dataset KC3 yang sama, maka penelitian ini dapat menghasilkan nilai *balance* yang lebih rendah, yaitu dengan penurunan *balance* sebesar 12,27%. Ada pun perbandingan nilai *balance* pada penelitian ini dengan penelitian sebelumnya, dapat dilihat di Gambar 4.17.



Gambar 4.17 Perbandingan Nilai *Pd, Balance, Pf* di KC3 Antara Metode Usulan dengan *EBD* + *CBC* Tanpa Seleksi Fitur

4.3.3. Hasil Skenario Pengujian di MW1

- *Balance* MW1



Gambar 4.18 Nilai *Balance* Pada Skenario Pengujian di MW1

Berdasarkan Gambar 4.18, *trend* nilai *balance* di semua seleksi fitur meningkat di *top* 6 fitur hingga *top* 31 fitur. Selain itu, *trend* menunjukkan bahwa nilai *balance* metode seleksi fitur *IG* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *balance* tertinggi dari setiap seleksi fitur yaitu *GR* 31 fitur menghasilkan *balance* 53,04%, *IG* 26 fitur menghasilkan *balance*

62,9%, *OR* 19 fitur menghasilkan *balance* 42,22%, *RFF* 6 fitur menghasilkan *balance* 59,14%, dan *SU* 19 fitur menghasilkan *balance* 53,62%. Adapun detil nilai *balance*, disajikan pada Tabel 4.18.

Tabel 4.18 Nilai *Balance* Pada MW1

Balance (MW1)					
Top Fitur	GR	IG	OR	RFF	SU
1	35,23139	41,92147	39,96143	36,00066	35,23139
2	36,23494	41,92147	39,66391	45,456	36,23494
3	36,30635	44,64555	34,86748	45,3322	33,26364
4	37,49946	54,51822	39,07423	45,26418	33,5625
5	35,58043	44,25613	39,07423	47,77162	35,58043
6	35,58043	46,4755	39,07423	**59,1363**	40,81479
7	40,81479	45,08576	35,00067	51,08005	38,79831
8	42,80211	47,3263	34,58517	57,6321	46,83467
9	38,76728	55,56997	34,84981	48,9816	47,11999
10	40,75052	47,40669	38,37175	46,54677	48,83869
11	46,87145	53,36686	39,9209	48,74103	48,71296
12	48,71296	51,01921	39,73848	48,89148	45,28265
13	48,93859	50,98183	38,34444	37,91875	45,38235
14	45,28265	53,30313	38,85141	43,14504	45,33142
15	45,45635	53,30313	38,78037	43,29158	45,45635
16	48,62319	53,43668	40,82592	43,75562	48,62319
17	39,31318	53,30313	38,9412	51,0134	39,31318
18	50,24807	53,2577	39,33487	51,0134	50,24807
19	41,24552	53,01781	**42,223**	51,26576	**53,6205**
20	51,11254	50,3275	40,41524	46,20332	50,95884
21	52,48028	57,52562	40,5513	46,70574	42,3542
22	51,8494	56,61355	40,38521	40,27605	51,8494
23	51,83307	57,69504	40,29887	40,43235	51,83307
24	50,29168	53,3234	40,29887	38,5265	50,29168
25	50,29168	53,33986	40,63354	30,92739	48,08959
26	48,41469	**62,9031**	40,63354	43,21285	48,08959
27	48,41469	59,12374	40,68345	43,38653	47,96872
28	48,29921	51,99317	40,59934	43,38876	47,96872
29	48,29921	57,76359	41,02803	42,6984	48,29921
30	48,29921	52,33536	40,89493	42,67786	53,03687
31	**53,0369**	55,45459	40,69366	42,85155	53,03687
32	53,03687	55,45459	40,69366	45,10387	53,03687
33	53,03687	55,31876	40,74356	49,67361	53,01152
34	53,01152	55,31876	40,9887	51,112	49,06766
35	49,06766	55,12269	40,64924	49,21168	49,06766
36	49,06766	55,12269	40,45803	49,21168	49,06766
37	49,06766	55,12269	39,98067	49,21168	49,06766

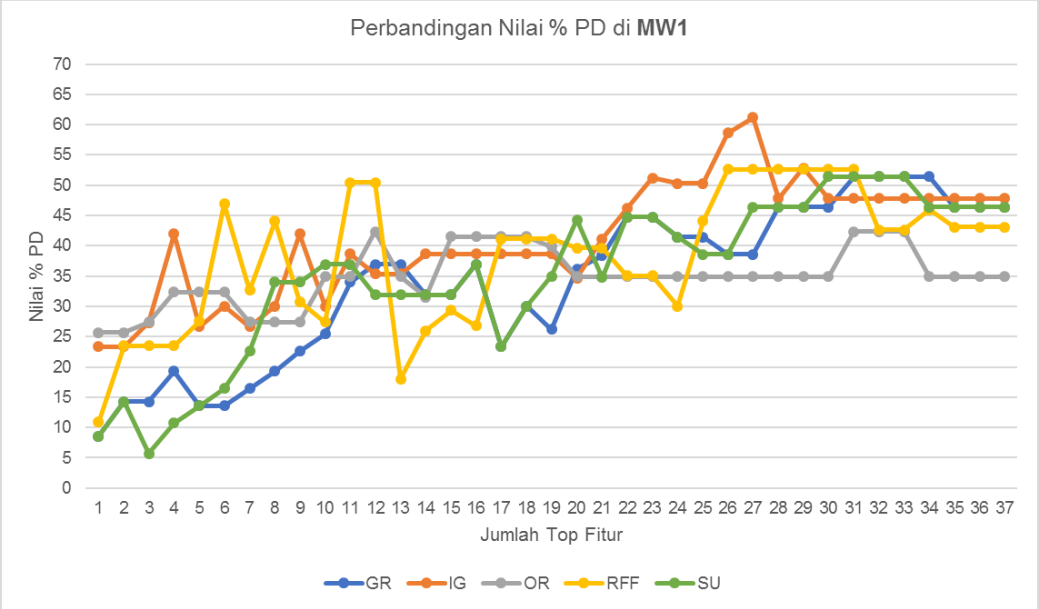
Berdasarkan Tabel 4.18, fitur terbaik MW1 berada pada *Top* 6 hingga 31 fitur. Jumlah *Top* fitur pada *OR* dan *SU* sama, yaitu 19. Jumlah *Top* fitur paling sedikit adalah *Top* fitur *RFF* yaitu 6, sedangkan jumlah *Top* fitur terbanyak adalah *GR* yaitu 31 fitur. Sedangkan *IG* dapat menyeleksi *Top* 26 fitur. Nilai *balance* yang berwarna kuning artinya nilai *balance* tertinggi di metode seleksi fitur tertentu. Sedangkan nilai *balance* yang berwarna biru artinya nilai *balance* tertinggi di antara

metode seleksi fitur lainnya. Pada Tabel 4.18 ditunjukkan bahwa hanya *Top* 26 fitur pada MW1 yang relevan dengan informasi untuk prediksi kesalahan perangkat lunak menggunakan *CBC*, yaitu *Top* fitur *IG*. Berdasarkan peringkat fitur *IG* di Tabel 4.18, maka 26 fitur tersebut disajikan pada Tabel 4.19.

Tabel 4.19 Fitur Terbaik Pada MW1

Peringkat	**No Fitur**	**Nama Fitur**
1	35	nl
2	29	Node_C
3	1	BLOC
4	13	Edge_C
5	25	V
6	14	ev(G)
7	18	I
8	37	LOC
9	33	n2
10	32	N1
11	22	L
12	21	B
13	16	SLOC
14	3	Call_C
15	31	N2
16	11	iv(G)
17	5	CLOC
18	27	Mod_Cond_C
19	2	Branch_C
20	6	Cond_C
21	24	T
22	20	E
23	28	Mul_Cond_C
24	9	Dec_C
25	4	C&SLOC
26	7	v(G)

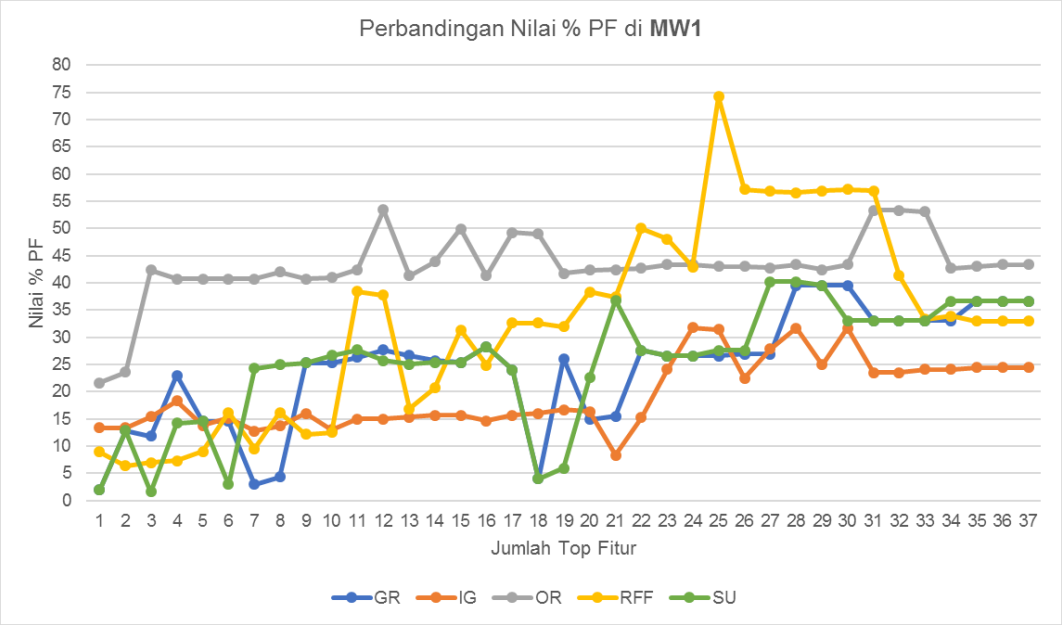
- *Pd* MW1



Gambar 4.19 Nilai *Pd* Pada Skenario Pengujian di MW1

Berdasarkan Gambar 4.19, *trend* nilai *pd* di semua seleksi fitur meningkat di *top* 12 fitur hingga *top* 31 fitur. Selain itu, *trend* menunjukkan bahwa nilai *pd* metode seleksi fitur *IG* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *pd* tertinggi dari setiap seleksi fitur yaitu *GR* 31 fitur menghasilkan *pd* 51,43%, *IG* 27 fitur menghasilkan *pd* 61,17%, *OR* 12 fitur menghasilkan *pd* 42,38%, *RFF* 26 fitur menghasilkan *pd* 52,62%, dan *SU* 30 fitur menghasilkan *pd* 51,43%.

- *Pf* MW1

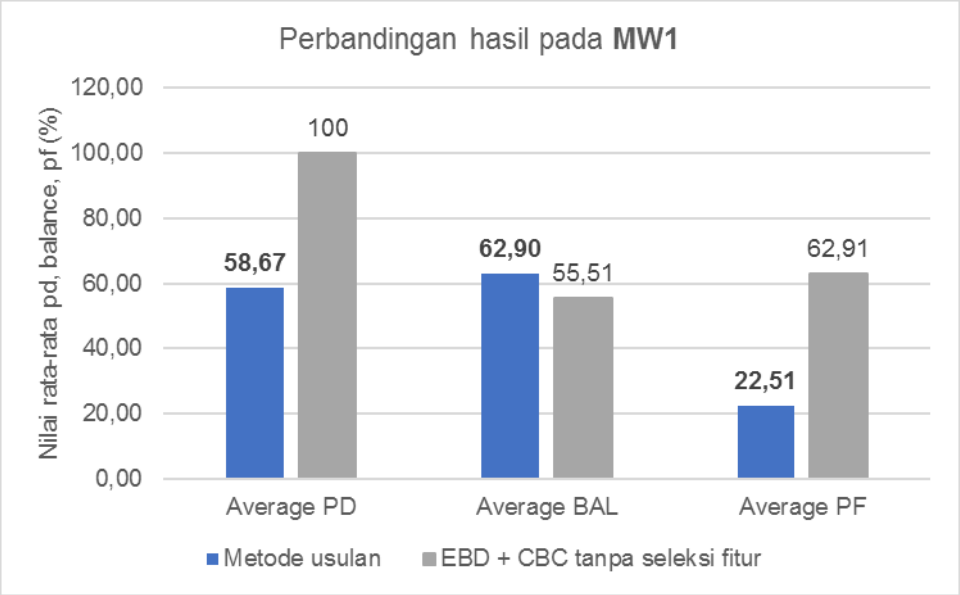


Gambar 4.20 Nilai *Pf* Pada Skenario Pengujian di MW1

Berdasarkan Gambar 4.20, nilai *pf* dari semua metode seleksi fitur cenderung menurun ketika *top* fitur dikurangi. Hanya metode seleksi fitur *RFF* yang menghasilkan nilai *pf* tertinggi yaitu pada *top* 25 fitur menghasilkan 74,2%.

- Perbandingan Hasil Penelitian di MW1

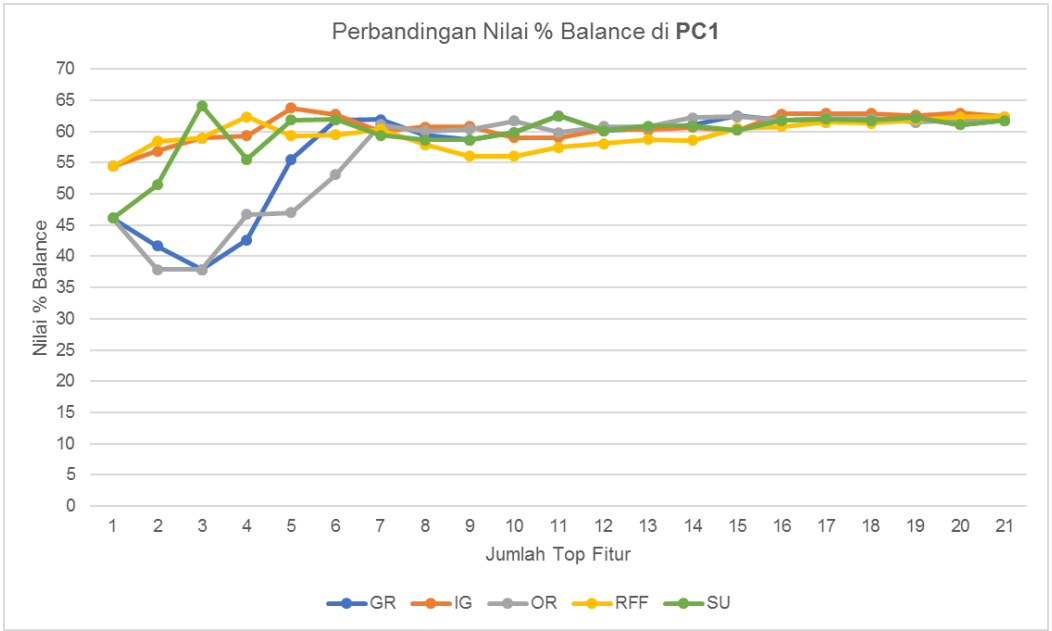
Jika dibandingkan dengan penelitian sebelumnya, pada dataset MW1 yang sama, maka penelitian ini dapat menghasilkan nilai *balance* yang lebih baik, yaitu dengan peningkatan *balance* sebesar 7,39%. Ada pun perbandingan nilai *balance* pada penelitian ini dengan penelitian sebelumnya, dapat dilihat di Gambar 4.21.



Gambar 4.21 Perbandingan Nilai *Pd, Balance, Pf* di MW1 Antara Metode Usulan dengan *EBD* + *CBC* Tanpa Seleksi Fitur

4.3.4. Hasil Skenario Pengujian di PC1

- *Balance* PC1



Gambar 4.22 Nilai *Balance* Pada Skenario Pengujian di PC1

Berdasarkan Gambar 4.22, *trend* nilai *balance* di semua seleksi fitur meningkat di *top* 3 fitur hingga *top* 20 fitur. Selain itu, *trend* menunjukkan bahwa nilai *balance* metode seleksi fitur *SU* cenderung lebih tinggi dari metode seleksi

fitur yang lain. Adapun rincian nilai *balance* tertinggi dari setiap seleksi fitur yaitu *GR* 15 fitur menghasilkan *balance* 62,5%, *IG* 5 fitur menghasilkan *balance* 63,78%, *OR* 15 fitur menghasilkan *balance* 62,35%, *RFF* 20 fitur menghasilkan *balance* 62,36%, dan *SU* 3 fitur menghasilkan *balance* 64,15%. Adapun detil nilai *balance*, disajikan pada Tabel 4.20.

Tabel 4.20 Nilai *Balance* Pada PC1

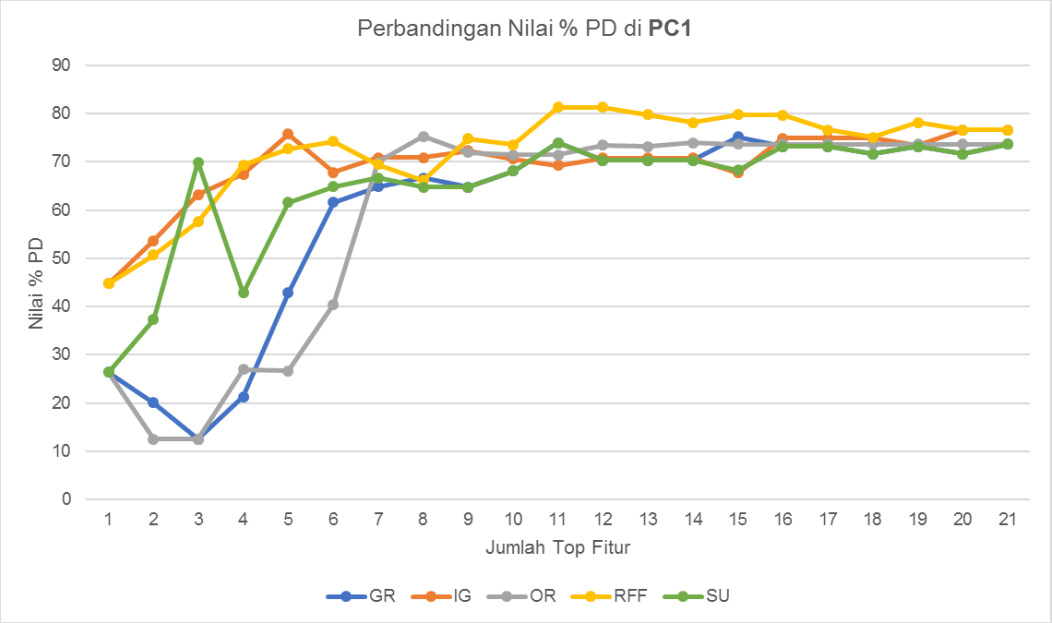
Balance (PC1)					
Top Fitur	GR	IG	OR	RFF	SU
1	46,11185	54,42231	46,11185	54,42231	46,11185
2	41,67639	56,85666	37,86314	58,39905	51,46871
3	37,86042	58,95072	37,86042	58,95749	**64,1497**
4	42,65763	59,29964	46,66611	62,33301	55,55135
5	55,55135	**63,7794**	47,02536	59,27717	61,84152
6	61,84152	62,7543	53,05721	59,48921	61,88077
7	61,88077	60,07351	61,10004	60,37134	59,39042
8	59,39042	60,68392	60,06861	57,9087	58,66588
9	58,66588	60,82726	60,33262	56,05725	58,66588
10	59,79185	59,02971	61,69768	56,05613	59,79185
11	62,49446	59,02908	59,79408	57,44633	62,49446
12	60,14557	60,38259	60,75617	58,02748	60,14557
13	60,52091	60,27843	60,74551	58,74328	60,85528
14	61,03154	60,63296	62,17364	58,55078	60,89184
15	**62,5028**	60,20161	**62,3543**	60,46364	60,19944
16	61,78923	62,76696	61,84006	60,80889	61,78923
17	61,94789	62,86785	62,09806	61,47973	61,94789
18	61,74638	62,86785	62,0886	61,29892	61,74638
19	62,22711	62,54765	61,48576	61,79968	62,22711
20	61,12229	62,94113	61,75104	**62,362**	61,12229
21	61,74587	62,362	61,74587	62,362	61,74587

Berdasarkan Tabel 4.20, fitur terbaik PC1 berada pada *Top* 3 hingga 20 fitur. Jumlah *Top* fitur pada *GR* dan *OR* sama, yaitu 15. Jumlah *Top* fitur paling sedikit adalah *Top* fitur *SU* yaitu 3, sedangkan jumlah *Top* fitur terbanyak adalah *RFF* yaitu 20 fitur. Sedangkan *IG* dapat menyeleksi *Top* 5 fitur. Nilai *balance* yang berwarna kuning artinya nilai *balance* tertinggi di metode seleksi fitur tertentu. Sedangkan nilai *balance* yang berwarna biru artinya nilai *balance* tertinggi di antara metode seleksi fitur lainnya. Pada Tabel 4.20 ditunjukkan bahwa hanya *Top* 3 fitur pada PC1 yang relevan dengan informasi untuk prediksi kesalahan perangkat lunak menggunakan *CBC*, yaitu *Top* fitur *SU*. Berdasarkan peringkat fitur *SU* di Tabel 4.20, maka 3 fitur tersebut disajikan pada Tabel 4.21.

Tabel 4.21 Fitur Terbaik Pada PC1

Peringkat	No Fitur	Nama Fitur
1	14	CLOC
2	3	ev(G)
3	15	C&SLOC

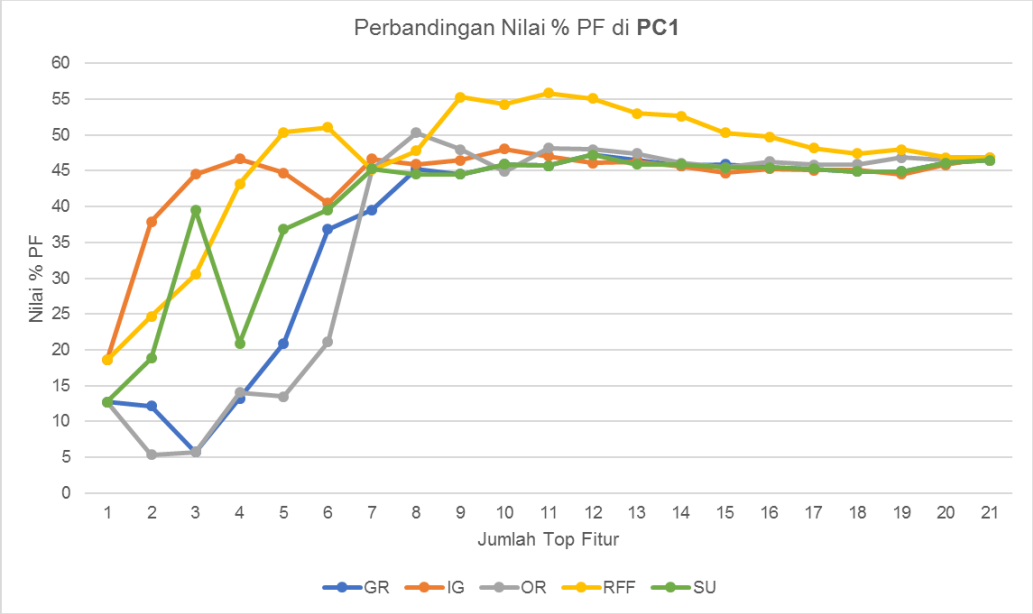
- *Pd* PC1



Gambar 4.23 Nilai *Pd* Pada Skenario Pengujian di PC1

Berdasarkan Gambar 4.23, *trend* nilai *pd* di semua seleksi fitur meningkat di *top* 8 fitur hingga *top* 20 fitur. Selain itu, *trend* menunjukkan bahwa nilai *pd* metode seleksi fitur *RFF* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *pd* tertinggi dari setiap seleksi fitur yaitu *GR* 15 fitur menghasilkan *pd* 75,17%, *IG* 20 fitur menghasilkan *pd* 76,6%, *OR* 8 fitur menghasilkan *pd* 75,32%, *RFF* 12 fitur menghasilkan *pd* 81,35%, dan *SU* 11 fitur menghasilkan *pd* 73,93%.

- *Pf* PC1

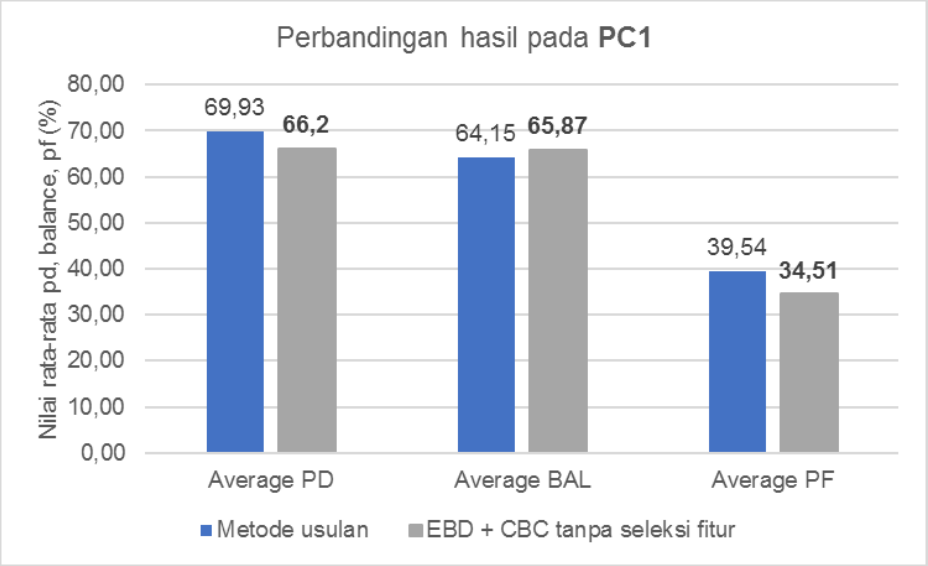


Gambar 4.24 Nilai *Pf* Pada Skenario Pengujian di PC1

Berdasarkan Gambar 4.24, nilai *pf* dari semua metode seleksi fitur cenderung menurun ketika *top* fitur dikurangi. Hanya metode seleksi fitur *RFF* yang menghasilkan nilai *pf* tertinggi yaitu pada *top* 11 fitur menghasilkan 55,87%.

- Perbandingan Hasil Penelitian di PC1

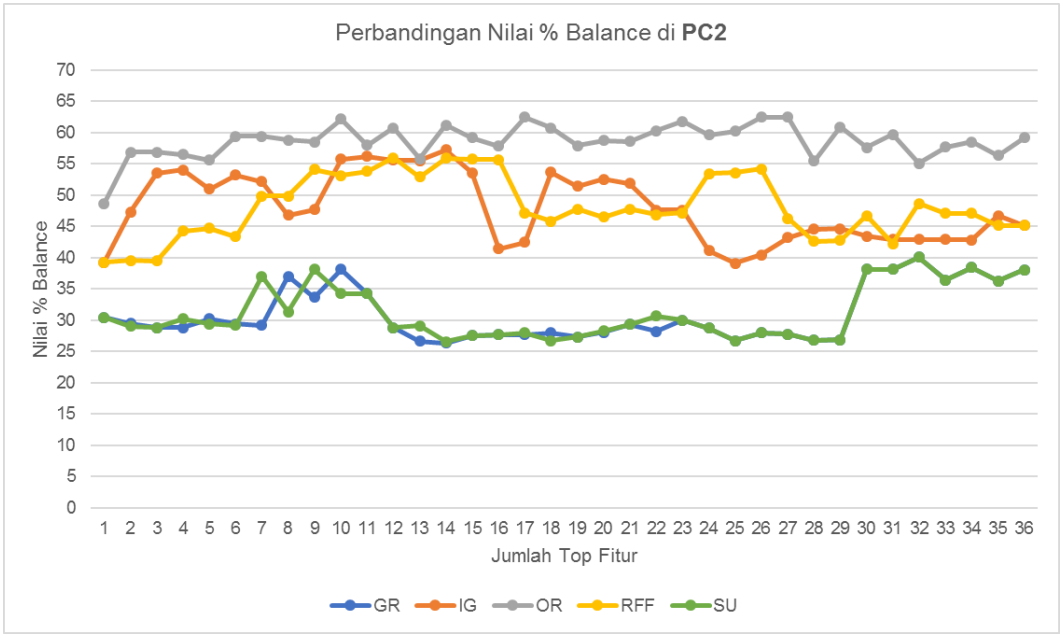
Jika dibandingkan dengan penelitian sebelumnya, pada dataset PC1 yang sama, maka penelitian ini dapat menghasilkan nilai *balance* yang lebih rendah, yaitu dengan penurunan *balance* sebesar 1,72%. Ada pun perbandingan nilai *balance* pada penelitian ini dengan penelitian sebelumnya, dapat dilihat di Gambar 4.25.



Gambar 4.25 Perbandingan Nilai *Pd*, *Balance*, *Pf* di PC1 Antara Metode Usulan dengan EBD + CBC Tanpa Seleksi Fitur

4.3.5. Hasil Skenario Pengujian di PC2

- *Balance* PC2



Gambar 4.26 Nilai *Balance* Pada Skenario Pengujian di PC2

Berdasarkan Gambar 4.26, *trend* nilai *balance* di semua seleksi fitur meningkat di *top* 12 fitur hingga *top* 32 fitur. Selain itu, *trend* menunjukkan bahwa nilai *balance* metode seleksi fitur *OR* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *balance* tertinggi dari setiap seleksi fitur yaitu

GR 32 fitur menghasilkan *balance* 40,09%, *IG* 14 fitur menghasilkan *balance* 57,2%, *OR* 26 fitur menghasilkan *balance* 62,48%, *RFF* 12 fitur menghasilkan *balance* 55,95%, dan *SU* 32 fitur menghasilkan *balance* 40,09%. Adapun detil nilai *balance*, disajikan pada Tabel 4.22.

Tabel 4.22 Nilai *Balance* Pada PC2

Balance (PC2)					
Top Fitur	GR	IG	OR	RFF	SU
1	30,40374	39,2295	48,62471	39,2295	30,40374
2	29,49107	47,29582	56,84065	39,55728	29,02754
3	28,77997	53,49313	56,84065	39,43795	28,77997
4	28,77997	54,00153	56,45726	44,23503	30,21576
5	30,21576	50,93446	55,60654	44,6926	29,4019
6	29,4019	53,23733	59,39677	43,37947	29,16079
7	29,16079	52,16426	59,39677	49,82545	36,99984
8	36,99984	46,80263	58,77496	49,8776	31,31051
9	33,66665	47,70018	58,50829	54,13158	38,11036
10	38,11036	55,74078	62,17671	53,11549	34,21048
11	34,21048	56,15498	58,00826	53,81232	34,21048
12	28,78206	55,57878	60,74181	**55,9513**	28,78206
13	26,64273	55,49951	55,84372	52,88597	29,10958
14	26,32305	**57,1993**	61,16158	55,87006	26,54947
15	27,50382	53,59888	59,206	55,75794	27,50382
16	27,65848	41,41677	57,82435	55,68125	27,65848
17	27,65848	42,46878	62,43398	47,16439	27,94056
18	27,94056	53,68029	60,74822	45,80688	26,70906
19	27,32973	51,37649	57,90953	47,75731	27,32973
20	28,03457	52,5119	58,75986	46,47363	28,24743
21	29,31916	51,88588	58,54623	47,78664	29,31916
22	28,20947	47,58656	60,31192	46,84446	30,6399
23	29,95329	47,58656	61,77502	47,19276	29,95329
24	28,69395	41,1375	59,59123	53,4234	28,69395
25	26,68996	39,10794	60,18368	53,5887	26,68996
26	28,00901	40,40576	**62,48**	54,16352	28,00901
27	27,77428	43,16935	62,47996	46,28406	27,77428
28	26,78436	44,55577	55,43458	42,61852	26,78436
29	26,83234	44,61806	60,85544	42,72437	26,83234
30	38,12148	43,39817	57,63413	46,72157	38,12148
31	38,09645	42,86803	59,67965	42,26366	38,09645
32	**40,0889**	42,86803	55,03446	48,62031	**40,0889**
33	36,43287	42,86803	57,644	47,11782	36,43287
34	38,40704	42,79416	58,49233	47,11782	38,40704
35	36,27648	46,73025	56,37321	45,16849	36,27648
36	38,07656	45,16849	59,19856	45,16849	38,07656

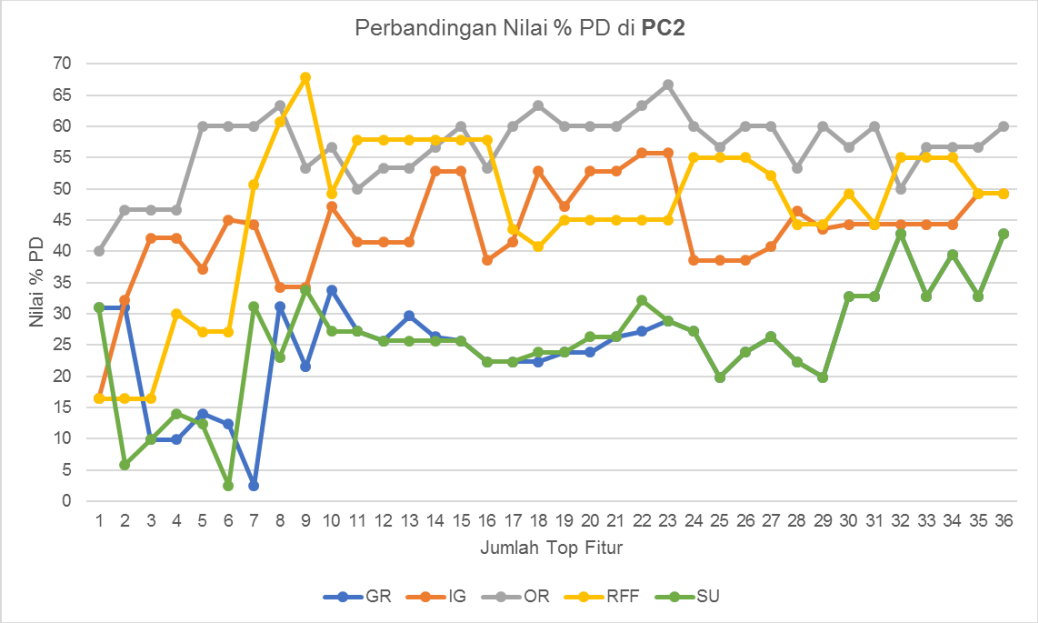
Berdasarkan Tabel 4.22, fitur terbaik PC2 berada pada *Top* 12 hingga 32 fitur. Jumlah *Top* fitur pada *GR* dan *SU* sama, yaitu 32. Jumlah *Top* fitur paling sedikit adalah *Top* fitur *RFF* yaitu 12. *IG* dapat menyeleksi *Top* 14 fitur. Sedangkan *OR* menyeleksi 26 fitur. Nilai *balance* yang berwarna kuning artinya nilai *balance*

tertinggi di metode seleksi fitur tertentu. Sedangkan nilai *balance* yang berwarna biru artinya nilai *balance* tertinggi di antara metode seleksi fitur lainnya. Pada Tabel 4.22 ditunjukkan bahwa hanya *Top* 26 fitur pada PC2 yang relevan dengan informasi untuk prediksi kesalahan perangkat lunak menggunakan *CBC*, yaitu *Top* fitur *OR*. Berdasarkan peringkat fitur *OR* di Tabel 4.22, maka 26 fitur tersebut disajikan pada Tabel 4.23.

Tabel 4.23 Fitur Terbaik Pada PC2

Peringkat	**No Fitur**	**Nama Fitur**
1	10	iv(G)
2	15	SLOC
3	35	%comment
4	16	Parameter_C
5	4	CLOC
6	23	T
7	19	E
8	3	C&SLOC
9	36	LOC
10	34	nl
11	31	N1
12	17	I
13	24	V
14	33	n1
15	32	n2
16	30	N2
17	11	id(G)
18	20	B
19	12	Edge_C
20	28	Node_C
21	21	L
22	1	Branch_C
23	18	D
24	25	Mainsev
25	6	v(G)
26	7	vd(G)

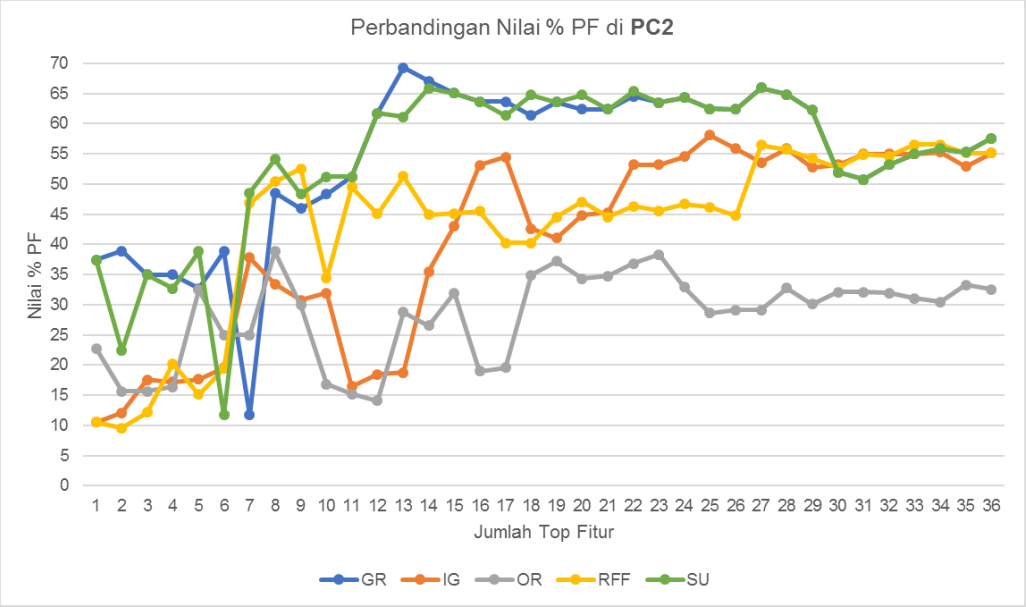
- *Pd* PC2



Gambar 4.27 Nilai *Pd* Pada Skenario Pengujian di PC2

Berdasarkan Gambar 4.27, *trend* nilai *pd* di semua seleksi fitur meningkat di *top* 11 fitur hingga *top* 32 fitur. Selain itu, *trend* menunjukkan bahwa nilai *pd* metode seleksi fitur *OR* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *pd* tertinggi dari setiap seleksi fitur yaitu *GR* 32 fitur menghasilkan *pd* 42,83%, *IG* 22 fitur menghasilkan *pd* 55,71%, *OR* 23 fitur menghasilkan *pd* 66,67%, *RFF* 11 fitur menghasilkan *pd* 57,86%, dan *SU* 32 fitur menghasilkan *pd* 42,83%.

- *Pf* PC2



Gambar 4.28 Nilai *Pf* Pada Skenario Pengujian di PC2

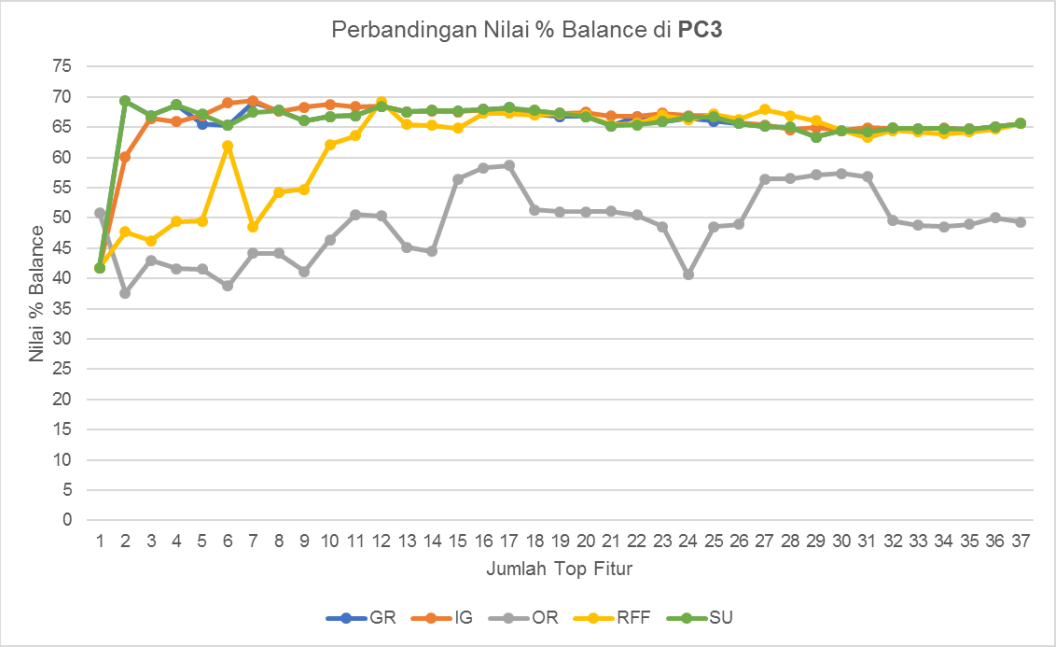
Berdasarkan Gambar 4.28, nilai *pf* dari semua metode seleksi fitur cenderung menurun ketika *top* fitur dikurangi. Hanya metode seleksi fitur *SU* yang menghasilkan nilai *pf* tertinggi yaitu pada *top* 27 fitur menghasilkan 65,93%.

- Perbandingan Hasil Penelitian di PC2

Dikarenakan terdapat perbedaan dimensi dataset PC2 dengan penelitian sebelumnya, maka pada hasil skenario pengujian ini tidak dapat dicantumkan perbandingan hasil *balance* di dataset PC2.

4.3.6. Hasil Skenario Pengujian di PC3

- *Balance* PC3



Gambar 4.29 Nilai *Balance* Pada Skenario Pengujian di PC3

Berdasarkan Gambar 4.29, *trend* nilai *balance* di semua seleksi fitur meningkat di *top* 2 fitur hingga *top* 17 fitur. Selain itu, *trend* menunjukkan bahwa nilai *balance* metode seleksi fitur *IG* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *balance* tertinggi dari setiap seleksi fitur yaitu *GR* 2 fitur menghasilkan *balance* 69,35%, *IG* 7 fitur menghasilkan *balance* 69,37%, *OR* 17 fitur menghasilkan *balance* 58,64%, *RFF* 12 fitur menghasilkan *balance* 69,26%, dan *SU* 2 fitur menghasilkan *balance* 69,35%. Dikarenakan perbedaan dimensi dataset PC3 dengan penelitian sebelumnya, maka tidak ada perbandingan hasil *balance* di dataset PC3. Adapun detil nilai *balance*, disajikan pada Tabel 4.24.

Tabel 4.24 Nilai *Balance* Pada PC3

Balance (PC3)					
Top Fitur	GR	IG	OR	RFF	SU
1	41,77286	41,77286	50,85012	41,77286	41,77286
2	**69,3477**	60,1522	37,56068	47,67519	**69,3477**
3	66,94003	66,47045	42,96646	46,24999	66,94003
4	68,72264	65,93855	41,58377	49,43145	68,72264
5	65,57213	66,92986	41,50908	49,50921	67,16924
6	65,34168	69,04861	38,75801	62,01583	65,34168
7	69,15009	**69,3692**	44,19082	48,5052	67,45509
8	67,79152	67,63723	44,19082	54,25784	67,79152
9	66,07043	68,29442	41,1314	54,71952	66,07043
10	66,75048	68,74943	46,41638	62,14145	66,75048
11	66,92575	68,36205	50,56315	63,62385	66,92575
12	68,46851	68,46851	50,36474	**69,2621**	68,46851
13	67,54329	67,54329	45,158	65,49286	67,54329
14	67,8134	67,60961	44,45581	65,26749	67,8134
15	67,68178	67,531	56,43535	64,84696	67,68178
16	67,9209	67,9209	58,27202	67,35052	67,9209
17	68,26604	68,00113	**58,6441**	67,34108	68,26604
18	67,15249	67,79856	51,29728	67,02776	67,79856
19	66,76687	67,33076	50,99119	67,21721	67,33076
20	66,75627	67,45001	51,05303	66,97495	66,75627
21	65,22297	66,84118	51,10042	65,40062	65,22297
22	66,73736	66,75422	50,46379	65,58725	65,34681
23	65,9378	67,33425	48,56139	66,99341	65,9378
24	66,50347	66,96181	40,63211	66,21823	66,65896
25	65,97426	66,96181	48,58407	67,14925	66,65896
26	65,6064	65,73411	48,95217	66,34711	65,6064
27	65,13012	65,39015	56,46275	67,90055	65,13012
28	65,01734	64,61619	56,50016	66,93864	65,01734
29	63,38425	64,94	57,11987	66,06415	63,38425
30	64,45706	64,45706	57,33708	64,45971	64,45706
31	64,22672	64,93886	56,79478	63,33595	64,22672
32	64,81048	64,81048	49,53802	64,45242	64,81048
33	64,75884	64,75884	48,75831	64,2365	64,75884
34	64,74331	64,83066	48,55203	63,88752	64,74331
35	64,6831	64,6831	48,92375	64,23165	64,6831
36	65,08417	65,08417	50,03316	64,71935	65,08417
37	65,61632	65,61632	49,35834	65,61632	65,61632

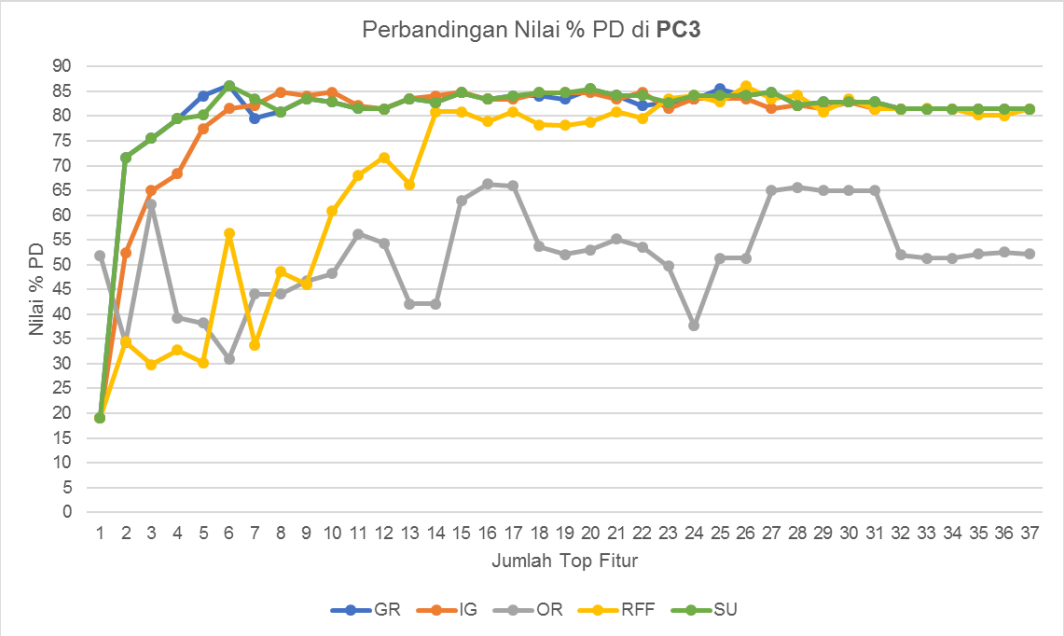
Berdasarkan Tabel 4.24, fitur terbaik PC3 berada pada *Top* 2 hingga 17 fitur. Jumlah *Top* fitur pada *GR* dan *SU* sama, yaitu 2. *IG* dapat menyeleksi *Top* 7 fitur, *OR Top* 17 fitur, dan *RFF Top* 12 fitur. Nilai *balance* yang berwarna kuning artinya nilai *balance* tertinggi di metode seleksi fitur tertentu. Sedangkan nilai *balance* yang berwarna biru artinya nilai *balance* tertinggi di antara metode seleksi fitur lainnya. Pada Tabel 4.24 ditunjukkan bahwa hanya *Top* 7 fitur pada PC3 yang relevan dengan informasi untuk prediksi kesalahan perangkat lunak menggunakan

CBC, yaitu *Top* fitur *IG*. Berdasarkan peringkat fitur *IG* di Tabel 4.24, maka 7 fitur tersebut disajikan pada Tabel 4.25.

Tabel 4.25 Fitur Terbaik Pada PC3

Peringkat	No Fitur	Nama Fitur
1	1	BLOC
2	26	Mainsev
3	10	dd(G)
4	18	I
5	30	Normv(G)
6	36	%comment
7	9	Dec_C

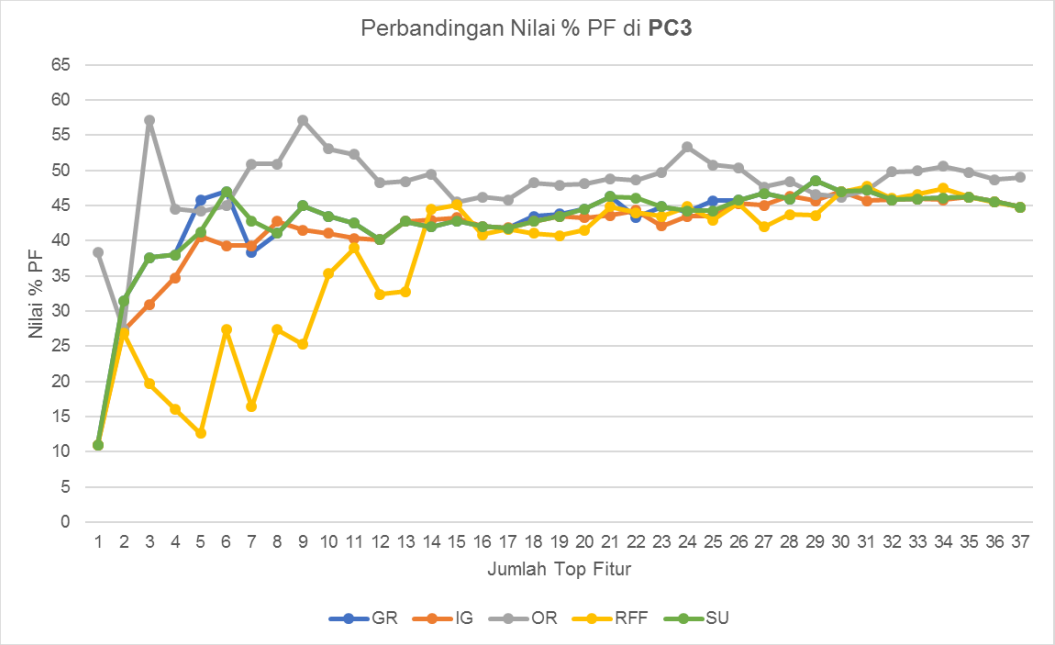
- *Pd* PC3



Gambar 4.30 Nilai *Pd* Pada Skenario Pengujian di PC3

Berdasarkan Gambar 4.30, *trend* nilai *pd* di semua seleksi fitur meningkat di *top* 6 fitur hingga *top* 26 fitur. Selain itu, *trend* menunjukkan bahwa nilai *pd* metode seleksi fitur *GR* dan *SU* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *pd* tertinggi dari setiap seleksi fitur yaitu *GR* 6 fitur menghasilkan *pd* 86,1%, *IG* 22 fitur menghasilkan *pd* 84,83%, *OR* 16 fitur menghasilkan *pd* 66,25%, *RFF* 26 fitur menghasilkan *pd* 86,08%, dan *SU* 6 fitur menghasilkan *pd* 86,1%. Dikarenakan perbedaan dimensi dataset PC3 dengan penelitian sebelumnya, maka tidak ada perbandingan hasil *pd* di dataset PC3.

- *Pf* PC3



Gambar 4.31 Nilai *Pf* Pada Skenario Pengujian di PC3

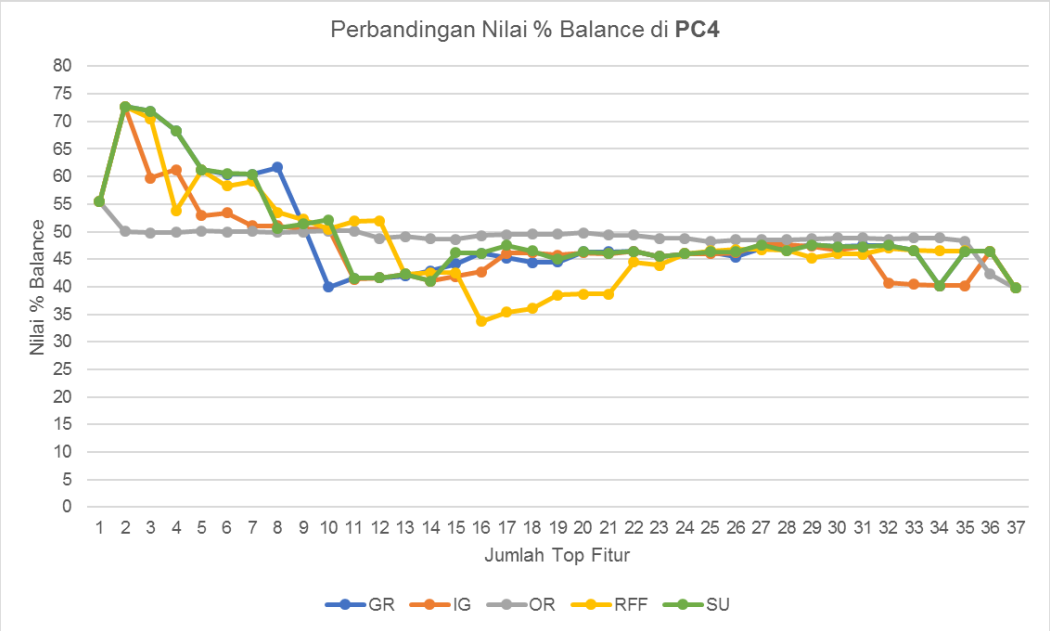
Berdasarkan Gambar 4.31, nilai *pf* dari semua metode seleksi fitur cenderung menurun ketika *top* fitur dikurangi. Hanya metode seleksi fitur *OR* yang menghasilkan nilai *pf* tertinggi yaitu pada *top* 3 fitur menghasilkan 57,17%.

- Perbandingan Hasil Penelitian di PC3

Dikarenakan terdapat perbedaan dimensi dataset PC3 dengan penelitian sebelumnya, maka pada hasil skenario pengujian ini tidak dapat dicantumkan perbandingan hasil *balance* di dataset PC3.

4.3.7. Hasil Skenario Pengujian di PC4

- *Balance* PC4



Gambar 4.32 Nilai *Balance* Pada Skenario Pengujian di PC4

Berdasarkan Gambar 4.32, *trend* nilai *balance* di semua seleksi fitur meningkat di *top* 1 fitur hingga *top* 2 fitur. Selain itu, *trend* menunjukkan bahwa nilai *balance* metode seleksi fitur *OR* cenderung lebih rendah dari metode seleksi fitur yang lain. Adapun rincian nilai *balance* tertinggi dari setiap seleksi fitur yaitu *GR* 2 fitur menghasilkan *balance* 72,71%, *IG* 2 fitur menghasilkan *balance* 72,71%, *OR* 1 fitur menghasilkan *balance* 55,44%, *RFF* 2 fitur menghasilkan *balance* 72,71%, dan *SU* 2 fitur menghasilkan *balance* 72,71%. Adapun detil nilai *balance*, disajikan pada Tabel 4.26.

Tabel 4.26 Nilai *Balance* Pada PC4

Balance (PC4)					
Top Fitur	GR	IG	OR	RFF	SU
1	55,43562	55,43562	**55,4356**	55,43562	55,43562
2	**72,7054**	**72,7054**	50,0887	**72,7054**	**72,7054**
3	71,8677	59,69421	49,77498	70,54502	71,8677
4	68,30415	61,23063	49,86626	53,75253	68,30415
5	61,22351	52,89234	50,17038	61,16909	61,22351
6	60,43575	53,37918	49,93436	58,30716	60,58417
7	60,43575	51,02643	50,05488	59,16604	60,43575
8	61,68594	51,02696	49,85339	53,49446	50,63415
9	51,43007	50,52861	49,96433	52,24221	51,43007
10	39,93063	50,44607	50,10903	50,44607	52,11549
11	41,53694	41,39181	50,10903	51,91012	41,53694
12	41,64402	41,64402	48,7641	51,97999	41,64402
13	42,00807	42,301	49,10207	42,301	42,301
14	42,90566	41,05928	48,68408	42,52212	41,05928
15	44,15729	41,87433	48,62014	42,54196	46,16428
16	46,05607	42,71174	49,26998	33,68798	46,05607
17	45,23751	46,07212	49,42517	35,43371	47,49667
18	44,40767	46,21455	49,52116	36,10743	46,53189
19	44,4901	45,77671	49,52116	38,48154	45,02128
20	46,36087	46,14623	49,79083	38,66863	46,36087
21	46,33769	45,99777	49,39396	38,66863	46,11711
22	46,40651	46,40651	49,39396	44,48505	46,40651
23	45,52381	45,52381	48,78018	43,90325	45,52381
24	45,97911	46,05093	48,78018	46,00885	46,05093
25	46,38046	46,04415	48,21985	46,47934	46,38046
26	45,38176	46,20966	48,52691	46,77351	46,38046
27	46,97315	47,56961	48,56517	46,77351	47,56961
28	46,97315	47,56961	48,56517	46,60185	46,51937
29	47,60725	47,3857	48,73287	45,28238	47,60725
30	47,28878	46,69794	48,89148	46,02169	47,28878
31	47,55791	47,28604	48,89213	45,95533	47,28878
32	47,55791	40,73571	48,58613	47,02286	47,55791
33	46,60329	40,46957	48,84308	46,57243	46,60329
34	40,19985	40,19985	48,85405	46,51858	40,19985
35	46,4476	40,19985	48,26521	46,51858	46,4476
36	46,4476	46,4476	42,30404	46,4476	46,4476
37	39,80988	39,80988	39,80988	39,80988	39,80988

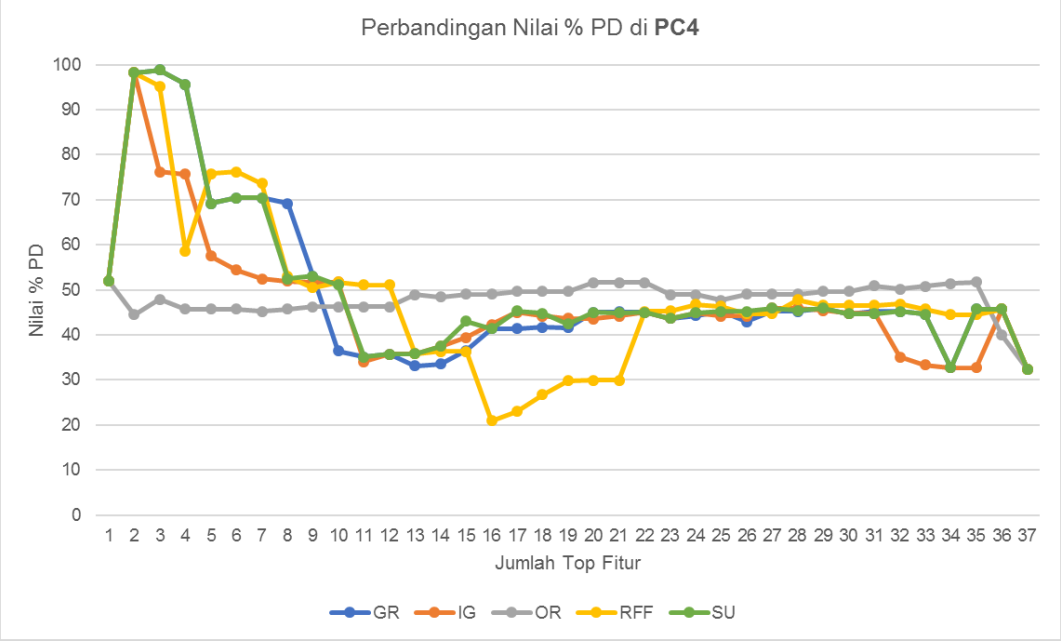
Berdasarkan Tabel 4.26, fitur terbaik PC4 berada pada *Top* 1 hingga 2 fitur. Jumlah *Top* fitur pada selain *OR* sama, yaitu 2. Sedangkan *OR* hanya menyeleksi *Top* 1 fitur. Nilai *balance* yang berwarna kuning artinya nilai *balance* tertinggi di metode seleksi fitur tertentu. Sedangkan nilai *balance* yang berwarna biru artinya nilai *balance* tertinggi di antara metode seleksi fitur lainnya. Pada Tabel 4.26 ditunjukkan bahwa hanya *Top* 2 fitur pada PC4 yang relevan dengan informasi untuk prediksi kesalahan perangkat lunak menggunakan *CBC*, yaitu *Top* fitur *GR*,

IG, *RFF*, dan *SU*. Berdasarkan peringkat fitur *IG* di Tabel 4.26, maka 2 fitur tersebut disajikan pada Tabel 4.27.

Tabel 4.27 Fitur Terbaik Pada PC4

Peringkat	No Fitur	Nama Fitur
1	4	C&SLOC
2	36	%comment

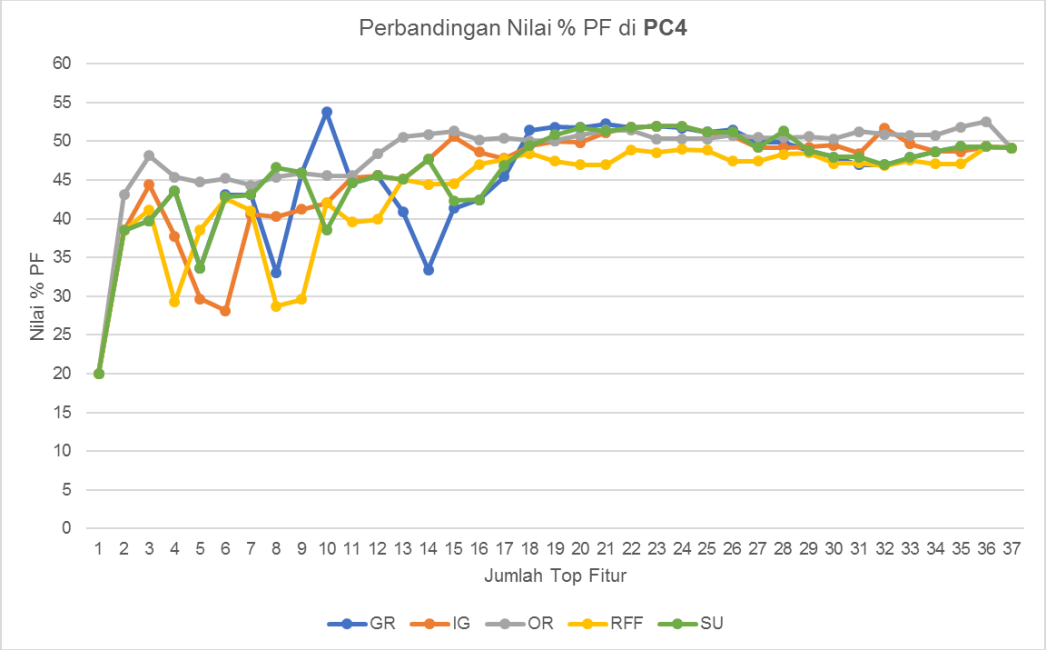
- *Pd* PC4



Gambar 4.33 Nilai *Pd* Pada Skenario Pengujian di PC4

Berdasarkan Gambar 4.33, *trend* nilai *pd* di semua seleksi fitur meningkat di *top* 1 fitur hingga *top* 3 fitur. Selain itu, *trend* menunjukkan bahwa nilai *pd* metode seleksi fitur *GR* dan *SU* cenderung lebih tinggi dari metode seleksi fitur yang lain. Adapun rincian nilai *pd* tertinggi dari setiap seleksi fitur yaitu *GR* 3 fitur menghasilkan *pd* 98,88%, *IG* 2 fitur menghasilkan *pd* 98,26%, *OR* 1 fitur menghasilkan *pd* 52,1%, *RFF* 2 fitur menghasilkan *pd* 98,26%, dan *SU* 3 fitur menghasilkan *pd* 98,88%.

- *Pf* PC4

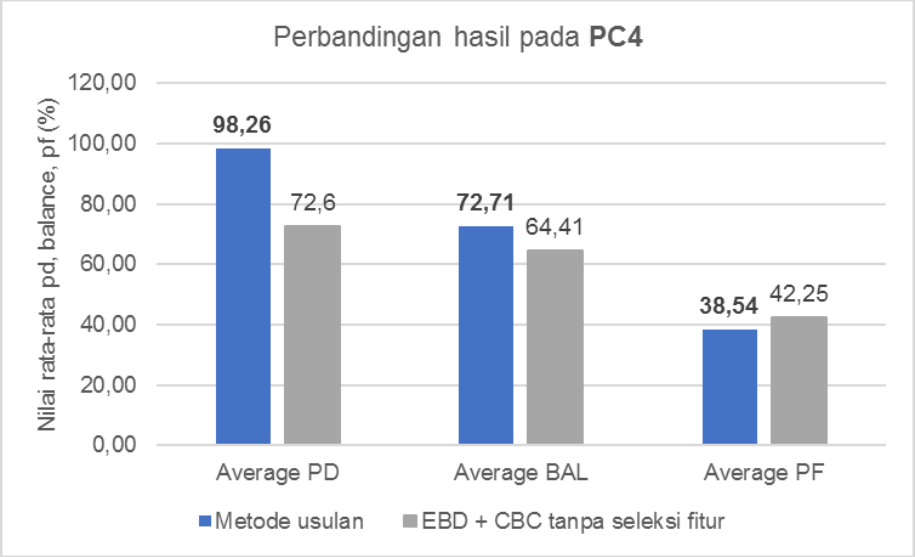


Gambar 4.34 Nilai *Pf* Pada Skenario Pengujian di PC4

Berdasarkan Gambar 4.34, nilai *pf* dari semua metode seleksi fitur cenderung menurun ketika *top* fitur dikurangi. Hanya metode seleksi fitur *RFF* yang menghasilkan nilai *pf* tertinggi yaitu pada *top* 10 fitur menghasilkan 53,76%.

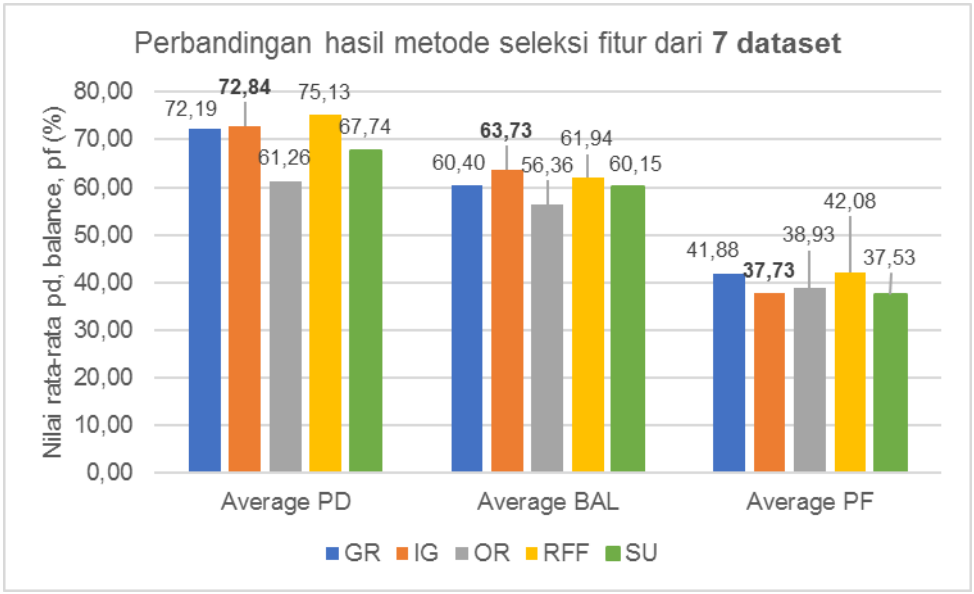
- Perbandingan Hasil Penelitian di PC4

Jika dibandingkan dengan penelitian sebelumnya, pada dataset PC4 yang sama, maka penelitian ini dapat menghasilkan nilai *balance* yang lebih baik, yaitu dengan peningkatan *balance* sebesar 8,30%. Ada pun perbandingan nilai *balance* pada penelitian ini dengan penelitian sebelumnya, dapat dilihat di Gambar 4.35.



Gambar 4.35 Perbandingan Nilai *Pd*, *Balance*, *Pf* di PC4 Antara Metode Usulan dengan *EBD* + *CBC* Tanpa Seleksi Fitur

4.3.8. Hasil Perbandingan Skenario Pengujian

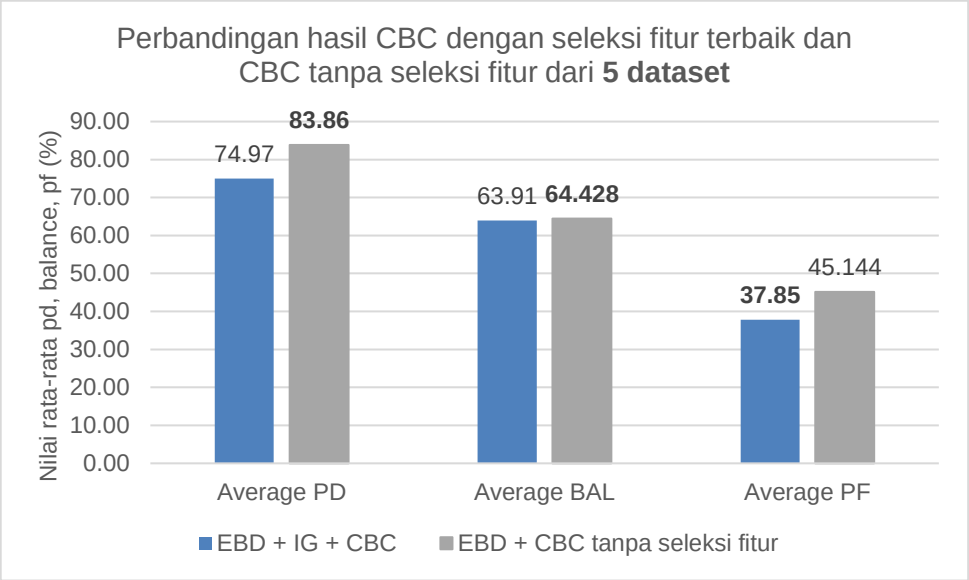


Gambar 4.36 Perbandingan Rata-Rata *Pd*, *Balance*, dan *Pf* dari 5 Metode Seleksi Fitur

Berdasarkan Gambar 4.36, seleksi fitur *IG* dapat menghasilkan nilai rata-rata *balance* paling tinggi dari 7 dataset, yaitu *balance* sebesar 63,73%. Adapun nilai *pd* tertinggi dihasilkan oleh metode seleksi fitur *RFF*, yaitu 75,13%. Namun, metode seleksi fitur *RFF* juga menghasilkan nilai *pf* paling tinggi, yaitu 42,08%.

Sedangkan nilai *pf* yang baik adalah yang mendekati angka 0. Oleh sebab itu, *balance RFF* tidak lebih baik dari *IG*.

Jadi, berdasarkan hasil uji coba, peningkatan nilai *balance* pada metode *CBC* dapat dicapai dengan melakukan pemeringkatan fitur terlebih dahulu. Kemudian dilakukan pemilihan sejumlah *Top X* fitur secara iteratif untuk menemukan nilai *balance* tertinggi di salah satu iterasi *Top X* fitur pada salah satu metode seleksi fitur. Adapun untuk menentukan metode seleksi fitur mana yang paling dapat meningkatkan nilai *balance* perlu dilakukan perbandingan hasil *balance* tertinggi di setiap metode seleksi fitur.



Gambar 4.37 Perbandingan Hasil dari 5 Dataset yang Sama Antara Metode Usulan dengan EBD + CBC Tanpa Seleksi Fitur

Berdasarkan Gambar 4.37, jika rata-rata *balance IG* dibandingkan dengan penelitian sebelumnya, dari 5 dataset yang sama (CM1, KC3, MW1, PC1, dan PC4), maka *balance* dari metode sebelumnya dapat menghasilkan nilai *balance* yang lebih baik, yaitu selisih *balance* sebesar 0,52%. Begitupun dengan nilai pd, terdapat selisih 8,89% yang artinya penelitian sebelumnya menghasilkan nilai *pd* yang lebih baik. Akan tetapi, jika dilihat dari nilai *pf*, metode penelitian sekarang dapat menghasilkan nilai *pf* yang lebih rendah yaitu dengan selisih 7,29%.

4.3.9. Resume Hasil Uji Coba

Berdasarkan hasil uji coba, hasil yang didapat adalah sebagai berikut:

1. Pada dataset CM1, nilai *balance* tertinggi diperoleh dari hasil kombinasi seleksi fitur *GR* dan klasifikasi *CBC*, yaitu *balance* sebesar 69,06%.
2. Jika hasil *balance* terbaik pada CM1 di penelitian ini dibandingkan dengan penelitian sebelumnya, maka penelitian ini dapat menghasilkan nilai *balance* 1,05% yang lebih baik.
3. Pada dataset KC3, nilai *balance* tertinggi diperoleh dari hasil kombinasi seleksi fitur *GR* dan klasifikasi *CBC*, seleksi fitur SU dan klasifikasi CBC, yaitu *balance* keduanya sebesar 56,07%.
4. Jika hasil *balance* terbaik pada KC3 di penelitian ini dibandingkan dengan penelitian sebelumnya, maka penelitian ini dapat menghasilkan nilai *balance* 12,27% yang lebih rendah.
5. Pada dataset MW1, nilai *balance* tertinggi diperoleh dari hasil kombinasi seleksi fitur *IG* dan klasifikasi *CBC*, yaitu *balance* sebesar 62,90%.
6. Jika hasil *balance* terbaik pada MW1 di penelitian ini dibandingkan dengan penelitian sebelumnya, maka penelitian ini dapat menghasilkan nilai *balance* 7,39% yang lebih baik.
7. Pada dataset PC1, nilai *balance* tertinggi diperoleh dari hasil kombinasi seleksi fitur *SU* dan klasifikasi *CBC*, yaitu *balance* sebesar 64,15%.
8. Jika hasil *balance* terbaik pada PC1 di penelitian ini dibandingkan dengan penelitian sebelumnya, maka penelitian ini dapat menghasilkan nilai *balance* 1,72% yang lebih rendah.
9. Pada dataset PC2, nilai *balance* tertinggi diperoleh dari hasil kombinasi seleksi fitur *OR* dan klasifikasi *CBC*, yaitu *balance* sebesar 62,48%.
10. Pada dataset PC3, nilai *balance* tertinggi diperoleh dari hasil kombinasi seleksi fitur *IG* dan klasifikasi *CBC*, yaitu *balance* sebesar 69,36%.
11. Hasil penelitian PC2 dan PC3 tidak dapat dibandingkan dengan metode sebelumnya, karena peneliti sebelumnya tidak menggunakan versi dataset PC2 dan PC3 yang sama, sehingga terdapat perbedaan jumlah fitur dari masing-masing dataset.

12. Pada dataset PC4, nilai *balance* tertinggi diperoleh dari hasil kombinasi seleksi fitur *GR* dan klasifikasi *CBC*, seleksi fitur *IG* dan klasifikasi *CBC*, seleksi fitur *RFF* dan klasifikasi *CBC*, seleksi fitur *SU* dan klasifikasi *CBC*, yaitu keempat *balance*-nya sebesar 72,71%.
13. Jika hasil *balance* terbaik pada PC4 di penelitian ini dibandingkan dengan penelitian sebelumnya, maka penelitian ini dapat menghasilkan nilai *balance* 8,30% yang lebih baik.
14. Jika dilakukan perhitungan rata-rata *balance* dari ketujuh dataset, maka kombinasi seleksi fitur *IG* dan klasifikasi *CBC* menghasilkan nilai *balance* tertinggi yaitu 63,73%.
15. Namun, jika dilakukan perbandingan rata-rata *balance* di lima dataset yang sama dengan penelitian sebelumnya, maka penelitian ini menghasilkan nilai *balance* 0,52% yang lebih rendah.
16. Berdasarkan nilai rata-rata *balance*, dapat disimpulkan tujuan pada penelitian ini tidak dapat tercapai, karena hasil balance penelitian sebelumnya masih lebih tinggi 0,52%. Namun 3 dari 5 dataset yang sama, dapat dihasilkan nilai *balance* yang lebih baik, yaitu di dataset CM1, MW1, dan PC4.
17. Adapun dari top 5 fitur, fitur yang sering muncul yaitu C&SLOC, CLOC, SLOC, dan BLOC.
18. Kelemahan kombinasi *EBD* + *CBC* yaitu tingkat komputasi yang tinggi. Karena pada setiap iterasi *fold*, *CBC* dapat menghasilkan titik *cluster* baru, dimana semua data pada dataset akan dihitung jarak ke masing-masing titik *cluster*. Semakin besar dimensi dataset, maka semakin lama waktu yang digunakan untuk proses *CBC*.

[Halaman ini sengaja dikosongkan]

BAB 5
PENUTUP

Pada bab ini dijelaskan mengenai kesimpulan dan saran dari hasil uji coba dan analisis yang telah dilakukan.

5.1. Kesimpulan

Dari hasil penelitian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Seleksi fitur menggunakan pendekatan *filter* memungkinkan pemeringkatan fitur berdasarkan tingkat relevansinya terhadap label *class*. Dengan pemilihan *Top X* fitur secara iteratif, maka dapat diketahui fitur apa saja yang relevan dalam memprediksi kesalahan perangkat lunak menggunakan *CBC*. Semakin relevan fitur yang digunakan, maka semakin baik pula prediksi yang dihasilkan.
2. Metode seleksi fitur yang dapat mengembangkan hasil prediksi kesalahan perangkat lunak menggunakan CBC ditentukan dengan membandingkan nilai rata-rata *balance* terbaik yang diperoleh masing-masing metode seleksi fitur. Berdasarkan hasil uji coba, metode seleksi fitur *IG* adalah yang paling baik dalam meningkatkan nilai *balance CBC*.
3. Jumlah fitur terbaik dapat diketahui dengan mengimplementasikan *CBC* di setiap iterasi *Top X* fitur. Nilai *balance* tertinggi di salah satu iterasi *Top X* fitur adalah jumlah fitur terbaik (relevan) yang dapat digunakan sebagai *input* dalam memprediksi kesalahan perangkat lunak.
4. Jika dibandingkan dengan integrasi metode *EBD* dan *CBC* tanpa seleksi fitur pada penelitian sebelumnya di lima dataset yang sama, maka metode usulan menghasilkan rata-rata *balance* 0,52% yang lebih rendah. Adapun rata-rata *balance* pada metode usulan sebesar 63,91% dan rata-rata *balance* pada penelitian sebelumnya sebesar 64,43%.
5. Akan tetapi, jika dibandingkan dengan metode sebelumnya (metode *EBD* dan *CBC* tanpa seleksi fitur), maka metode usulan dapat menghasilkan nilai balance yang lebih tinggi di 3 dataset dari total 5 dataset. Adapun selisih peningkatan

nilai *balance* di 3 dataset tersebut yaitu CM1 1,5%, MW1 7,39%, dan PC4 8,3%.

5.2. Saran

Berdasarkan hasil pengujian, untuk proses *EBD* perlu dicari nilai *dumping factor* sebagai *threshold* jumlah maksimal toleransi redundansi data biner dengan kelas yang berbeda. Selain itu, perlu diteliti lebih lanjut tentang penggunaan metode yang dapat mengatasi persebaran kelas yang tidak merata. Metode seleksi fitur dengan pendekatan *wrapper* juga perlu diteliti lebih lanjut untuk mengoptimalkan hasil kombinasi fitur yang akan digunakan *CBC*.

DAFTAR PUSTAKA

Abaei, G. & Selamat, A., 2014. A survey on software fault detection based on different prediction approaches. *Vietnam Journal of Computer Science*, 1(2), pp.79–95.

Abraham, R., Simha, J.B. & Iyengar, S.S., 2009. Effective Discretization and Hybrid feature selection using Naïve Bayesian classifier for Medical datamining. *International Journal of Computational Intelligence Research*, 5(2), pp.116–129.

Akalya Devi, C., Kannammal, K.E. & Surendiran, B., 2012. A Hybrid Feature Selection Model For Software Fault Prediction. *International Journal on Computational Sciences & Applications (IJCSA)*, 2(2), pp.25–35.

Akbar, M.S., 2017. *Prediksi Cacat Perangkat Lunak Dengan Optimasi Naive Bayes Menggunakan Gain Ratio*. Institut Teknologi Sepuluh Nopember.

Antony, D.A. et al., 2016. Software Fault Detection using Honey Bee Optimization. *International Journal of Applied Information Systems (IJAIS)*, 11(1), pp.1–9.

Arora, I., Tetarwal, V. & Saha, A., 2015. Open Issues in Software Defect Prediction. *Procedia - Procedia Computer Science*, 46(Icict 2014), pp.906–912. Available at: http://dx.doi.org/10.1016/j.procs.2015.02.161.

Catal, C., 2012. Performance evaluation metrics for software fault prediction studies. *Acta Polytechnica Hungarica*, 9(4), pp.193–206.

Catal, C., 2011. Software fault prediction: A literature review and current trends. *Expert Systems with Applications*, 38(4), pp.4626–4636. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0957417410011681 [Accessed October 19, 2016].

Catal, C. & Diri, B., 2009. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences*, 179(8), pp.1040–1058. Available at: http://dx.doi.org/10.1016/j.ins.2008.12.001.

Chandrashekar, G. & Sahin, F., 2014. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1), pp.16–28. Available at: http://dx.doi.org/10.1016/j.compeleceng.2013.11.024.

Dougherty, J., Kohavi, R. & Sahami, M., 1995. Supervised and unsupervised discretization of continuous features. *In Proceedings of the 12th international conference on machine learning*, pp.194–202. Available at: http://www.ncbi.nlm.nih.gov/pubmed/15003161%5Cnhttp://cid.oxfordjournals.org/lookup/doi/10.1093/cid/cir991%5Cnhttp://www.scielo.cl/pdf/udecada/v15n26/art06.pdf%5Cnhttp://www.scopus.com/inward/record.url?eid=2-s2.0-84861150233&partnerID=tZOtx3y1.

Erturk, E. & Sezer, E.A., 2015. A comparison of some soft computing methods for software fault prediction. *EXPERT SYSTEMS WITH APPLICATIONS*, 42(4), pp.1872–1879. Available at: http://dx.doi.org/10.1016/j.eswa.2014.10.025.

Fayyad, U.M. & Irani, K.B., 1993. Multi-Interval Discretization of Continuous-

Valued Attributes for Classification Learning.

Gao, K. et al., 2011. Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Software - Practice and Experience*, 39(7), pp.701–736.

Hall, T. et al., 2012. A Systematic Review of Fault Prediction Performance in Software Engineering. *IEEE Transactions on Software Engineering*, 38(6), pp.1276–1304.

Irawan, D.A., Baizal, Z.A. & Perdana, E.G., 2011. ANALISIS DAN IMPLEMENTASI ALGORITMA RELIEFF UNTUK FEATURE SELECTION PADA KLASIFIKASI DATASET MULTICLASS.

Karegowda, A.G., Manjunath, A.S. & Jayaram, M.A., 2010. Comparative Study of Attribute Selection Using Gain Ratio and Correlation Based Feature Selection. *International Journal of Information Technology and Knowledge Management*, 2(2), pp.271–277.

Kohavi, R. & Sahami, M., 1996. Error-Based and Entropy-Based Discretization of Continuous Features. *Journal of microscopy*, 237(3), pp.487–96. Available at: http://robotics.stanford.edu/users/sahami/papers-dir/kdd96-disc.pdf.

Kumar, A. & Zhang, D., 2007. Hand-geometry recognition using entropy-based discretization. *IEEE Transactions on Information Forensics and Security*, 2(2), pp.181–187.

Ladha, L. & Deepa, T., 2011. Feature selection methods and algorithms. *International Journal on …*, 3(5), pp.1787–1797. Available at: http://journals.indexcopernicus.com/abstract.php?icid=945099.

Lessmann, S. et al., 2008. Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, 34(4), pp.485–496. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4527256%5Cnhttp://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4527256.

Malhotra, R., 2015. A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing Journal*, 27, pp.504–518. Available at: http://dx.doi.org/10.1016/j.asoc.2014.11.023.

Menzies, T., Greenwald, J. & Frank, A., 2007. Data Mining Static Code Attributes to Learn Defect Predictors. *IEEE Transactions on Software Engineering*, 33(1), pp.2–14.

Moeyersoms, J. et al., 2015. Comprehensible software fault and effort prediction: A data mining approach. *Journal of Systems and Software*, 100, pp.80–90.

Murti, D.H., Suciati, N. & Nanjaya, D.J., 2005. Clustering data non-numerik dengan pendekatan algoritma k-means dan hamming distance studi kasus biro jodoh. *Jurnal Ilmiah Teknologi Informasi (JUTI)*, 4, pp.46–53.

Najadat, H. & Alsmadi, I., 2012. Enhance Rule Based Detection for Software Fault Prone Modules Computer Information Systems Department Computer Information Systems Department. *International Journal*, 6(1), pp.75–86.

Novakovic, J., 2010. The Impact of Feature Selection on the Accuracy of Naive Bayes Classifier. *18th Telecommunications forum TELFOR*, 2, pp.1113–1116.

Quinlan, J.R., 1986. Induction of Decision Trees. *Machine Learning*, 1(1), pp.81–106.

R.P.L.DURGABAI, 2014. Feature Selection using ReliefF Algorithm. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(10), pp.8215–8218.

Sathyaraj, R. & Prabu, S., 2015. An Approach for Software Fault Prediction to Measure the Quality of Different Prediction Methodologies using Software Metrics. *Indian Journal of Science and Technology*, 8(December).

Singh, D.A.A.G., Fernando, A.E. & Leavline, E.J., 2016. Experimental study on feature selection methods for software fault detection. *International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp.1–6.

Singh, P. & Verma, S., 2014. An Efficient Software Fault Prediction Model using Cluster based Classification. *International Journal of Applied Information Systems (IJAIS)*, 7(3), pp.35–41.

Singh, P. & Verma, S., 2009. An Investigation of the Effect of Discretization on Defect Prediction Using Static Measures. *International Conference on Advances in Computing, Control, and Telecommunication Technologies*, pp.837–839.

Singh, P. & Verma, S., 2015. Cross Project Software Fault Prediction at Design Phase. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 9(3), pp.800–805. Available at: http://dx.doi.org/10.5370/JEET.2014.9.4.742.

Singh, P. & Vyas, O.P., 2014. Software Fault Prediction Model for Embedded Software : A Novel finding. *International Journal of Computer Science and Information Technologies*, 5(2), pp.2348–2354.

Yang, F. et al., 2011. An Improved Feature Selection Approach Based on ReliefF and Mutual Information. *International Conference on Information Science and Technology*, pp.246–250.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

1.1 Dataset CM1

LOC	v(g)	ev(g)	iv(g)	N	V	L	D	I	E	B	T	IOCode	CLOC	BLOC	C&SLOC	n1	n2	N1	N2	Branch_C	faulty
1.1	1.4	1.4	1.4	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	2	2	2	2	1.2	1.2	1.2	1.2	1.4	F
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T
24	5	1	3	63	309.13	0.11	9.5	32.54	2936.77	0.1	163.15	1	0	6	0	15	15	44	19	9	F
20	4	4	2	47	215.49	0.06	16	13.47	3447.89	0.07	191.55	0	0	3	0	16	8	31	16	7	F
24	6	6	2	72	346.13	0.06	17.33	19.97	5999.58	0.12	333.31	0	0	3	0	16	12	46	26	11	F
24	6	6	2	72	346.13	0.06	17.33	19.97	5999.58	0.12	333.31	0	0	3	0	16	12	46	26	11	F
7	1	1	1	11	34.87	0.5	2	17.43	69.74	0.01	3.87	0	0	1	0	4	5	6	5	1	F
12	2	1	2	23	94.01	0.16	6.43	14.62	604.36	0.03	33.58	0	0	7	0	10	7	14	9	3	F
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
28	6	5	5	104	564.33	0.06	16.09	35.08	9078.38	0.19	504.35	2	7	0	0	20	23	67	37	11	T

1.2 Dataset KC3

LocBlank	Branch Count	CallPairs	LocCodeAndComment	LocComments	ConditionCount	CyclomaticComplexity	CyclomaticDensity	DecisionCount	DecisionDensity	Design Complexity	Design Density	EdgeCount	EssentialComplexity	EssentialDensity	LocExecutable	ParameterCount	GlobalDataComplexity	GlobalDataDensity	HalsteadContent	HalsteadDifficulty	HalsteadEffort	HalsteadErrorEst	HalsteadLength	HalsteadLevel	HalsteadProgTime	HalsteadVolume	MaintenanceSeverity	ModifiedConditionCount	MultipleConditionCount	NodeCount	NormalizedCylomaticComplexity	NumOperands	NumOperators	NumUniqueOperands	NumUniqueOperators	NumberOfLines	PercentComments	LocTotal	Faulty
0	1	1	0	0	0	1	0.33	0	0	1	1	2	1	0	3	1	1	1	9.51	3	85.59	0.01	9	0.33	4.75	28.53	1	0	0	3	0.25	3	6	3	6	4	0	3	F
1	5	4	0	0	8	3	0.27	4	2.0	3	1	13	1	0	11	1	3	1	23.27	11.38	3010.96	0.09	57	0.09	167.28	264.7	0.33	2	4	12	0.23	21	36	12	13	13	0	11	F
0	1	0	0	0	0	1	0.33	0	0	1	1	1	1	0	3	1	1	1	3.62	5	90.47	0.01	7	0.2	5.03	18.09	1	0	0	2	0.25	2	5	1	5	4	0	3	F
0	1	0	0	0	1	0.33	0	0	1	1	1	1	0	3	0	1	1	1	7.86	2.5	49.13	0.01	7	0.4	2.73	19.65	1	0	0	2	0.25	2	5	2	5	4	0	3	F
5	18	18	0	2	20	10	0.17	10	2.0	8	0.8	62	5	0.44	58	0	9	0.9	69.5	23.96	39892.13	0.56	269	0.04	2216.23	1665.06	0.5	5	10	54	0.15	92	177	48	25	66	3.33	58	F
0	1	2	0	0	0	1	0.17	0	0	1	1	3	1	0	6	0	1	1	21.8	6.19	834.61	0.04	33	0.16	46.37	134.89	1	0	0	4	0.14	11	22	8	9	7	0	6	F
1	1	2	0	0	0	1	0.17	0	0	1	1	3	1	0	6	3	1	1	26.01	7.86	1605.45	0.07	44	0.13	89.19	204.33	1	0	0	4	0.11	20	24	14	11	9	0	6	F
0	3	1	0	0	4	2	0.22	2	2.0	1	0.5	6	1	0	9	1	1	0.5	14.87	8.57	1092.32	0.04	30	0.12	60.68	127.44	0.5	1	2	6	0.2	10	20	7	12	10	0	9	F
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
22	43	20	3	11	38	26	0.2	18	2.11	18	0.69	132	15	0.56	128	0	19	0.73	109.63	42.03	193686.35	1.54	666	0.02	10760.35	4607.96	0.58	10	19	108	0.16	255	411	91	30	166	9.86	131	F

1.3 Dataset MW1

LocBlank	Branch Count	CallPairs	LocCodeAndComment	LocComments	ConditionCount	CyclomaticComplexity	CyclomaticDensity	DecisionCount	DecisionDensity	Design Complexity	Design Density	EdgeCount	EssentialComplexity	EssentialDensity	LocExecutable	ParameterCount	HalsteadContent	HalsteadDifficulty	HalsteadEffort	HalsteadErrorEst	HalsteadLength	HalsteadLevel	HalsteadProgTime	HalsteadVolume	MaintenanceSeverity	ModifiedConditionCount	MultipleConditionCount	NodeCount	NormalizedCylomaticComplexity	NumOperands	NumOperators	NumUniqueOperands	NumUniqueOperators	NumberOfLines	PercentComments	LocTotal	Faulty
2	1	0	0	3	0	1	0.17	0	0	1	1	1	1	0	6	1	14.09	5.4	410.98	0.03	22	0.19	22.83	76.11	1	0	0	2	0.08	9	13	5	6	12	33.33	6	F
3	5	4	0	4	8	3	0.25	4	2.0	3	1	11	1	0	12	2	42.23	6.94	2036.45	0.1	61	0.14	113.14	293.25	0.33	2	4	10	0.15	25	36	18	10	20	25	12	F
4	11	2	0	8	20	6	0.3	10	2.0	3	0.5	24	1	0	20	1	22.99	18.44	7817.84	0.14	82	0.05	434.32	423.93	0.17	5	10	20	0.18	33	49	17	19	33	28.57	20	F
2	7	4	0	1	12	4	0.25	6	2.0	4	1	19	4	1	16	0	42.95	4.15	741.1	0.06	42	0.24	41.17	178.41	1	3	6	17	0.2	18	24	13	6	20	5.88	16	F
1	1	3	0	4	0	1	0.14	0	0	1	1	6	1	0	7	0	44.57	3.46	534.07	0.05	37	0.29	29.67	154.29	1	0	0	7	0.08	18	19	13	5	13	36.36	7	F
1	5	4	0	0	8	3	0.27	4	2.0	3	1	13	1	0	11	0	31.66	6.43	1308.6	0.07	45	0.16	72.7	203.56	0.33	2	4	12	0.23	20	25	14	9	13	0	11	F
11	12	10	0	15	6	9	0.16	2	3.0	8	0.89	29	1	0	58	0	39.93	28.29	31947.46	0.38	170	0.04	1774.86	1129.46	0.11	2	3	22	0.11	72	98	56	44	85	20.55	58	F
2	1	2	13	4	0	1	0.06	0	0	1	1	15	1	0	3	1	104.67	3.73	1454.15	0.13	83	0.27	80.79	390.14	1	0	0	16	0.04	41	42	22	4	23	85	16	F
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
4	3	10	0	3	4	2	0.13	2	2.0	2	1	17	1	0	16	0	67.54	3.24	706.95	0.07	49	0.31	39.28	218.51	0.5	1	2	17	0.08	22	27	17	5	24	15.79	16	F

1.4 Dataset PC1

loc	v(g)	ev(g)	iv(G)	N	V	L	D	I	E	B	T	lOCode	lOComment	locCodeAndComment	lOBlank	uniq_Op	uniq_Opnd	total_Op	total_Opnd	branchCount	faulty
1.1	1.4	1.4	1.4	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	2	2	2	2	1.2	1.2	1.2	1.2	1.4	F
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T
91	9	3	2	318	2089.21	0.04	27.68	75.47	57833.24	0.7	3212.96	80	44	11	31	29	66	192	126	17	T
109	21	5	18	381	2547.56	0.04	28.37	89.79	72282.68	0.85	4015.7	97	41	12	24	28	75	229	152	38	T
505	106	41	82	2339	20696.93	0.01	75.93	272.58	1571506.88	6.9	87305.94	457	71	48	49	64	397	1397	942	178	T
107	25	7	14	619	4282.78	0.02	52.91	80.95	226588.75	1.43	12588.26	103	32	4	39	35	86	359	260	40	T
74	11	1	8	294	1917.93	0.03	28.77	66.66	55178.46	0.64	3065.47	60	71	14	49	29	63	169	125	21	T
602	136	123	123	2785	25942.69	0.01	105.26	246.47	2730637.23	8.65	151702.06	600	40	2	225	99	538	1641	1144	236	T
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
26	18	13	6	228	1335.62	0.03	35.81	37.29	47834.26	0.45	2657.46	26	0	0	1	23	35	119	109	35	F

1.5 Dataset PC2

BranchCount	CallPairs	LocCodeAndComment	LocComments	ConditionCount	CyclomaticComplexity	CyclomaticDensity	DecisionCount	DecisionDensity	DesignComplexity	DesignDensity	EdgeCount	EssentialComplexity	EssentialDensity	LocExecutable	ParameterCount	HalsteadContent	HalsteadDifficulty	HalsteadEffort	HalsteadErrorEst	HalsteadLength	HalsteadLevel	HalsteadProgTime	HalsteadVolume	MaintenanceSeverity	ModifiedConditionCount	MultipleConditionCount	NodeCount	NormalizedCylomaticComplexity	NumOperands	NumOperators	NumUniqueOperands	NumUniqueOperators	NumberOfLines	PercentComments	LocTotal	Faulty
1	0	0	0	0	1	1	0	0	1	1	1	1	0	0	2	5.33	1.5	12	0	4	0.67	0.67	8	1	0	0	2	0.5	1	3	1	3	2	0	0	F
1	1	0	0	0	1	1	0	0	1	1	2	1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	3	1	1	0	1	0	1	0	0	F
1	4	7	24	0	1	0.13	0	0	1	1	6	1	0	1	0	17.88	7.43	986.77	0.04	34	0.13	54.82	132.83	1	0	0	7	0.03	13	21	7	8	34	96.88	8	F
1	1	11	3	0	1	0.08	0	0	1	1	2	1	0	1	0	42.62	7.81	2598.31	0.11	77	0.13	144.35	332.79	1	0	0	3	0.06	29	48	13	7	17	93.33	12	F
1	1	0	0	0	1	1	0	0	1	1	2	1	0	1	3	33.44	0.63	13.06	0.01	9	1.6	0.73	20.9	1	0	0	3	0.33	5	4	4	1	3	0	1	F
1	1	0	0	0	1	1	0	0	1	1	2	1	0	0	1	4	0.5	1	0	2	2	0.06	2	1	0	0	3	1	1	1	1	1	1	0	0	F
1	0	1	0	0	1	0.5	0	0	1	1	1	1	0	1	2	10.87	3.5	133.19	0.01	11	0.29	7.4	38.05	1	0	0	2	0.33	4	7	4	7	3	50	2	F
3	1	1	0	6	2	1	2	3.0	1	0.5	5	1	0	1	1	11.07	4.38	211.89	0.02	14	0.23	11.77	48.43	0.5	2	3	5	0.5	5	9	4	7	4	50	2	F
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
3	2	2	0	4	2	0.67	2	2.0	2	1	6	1	0	1	0	12.18	3	109.62	0.01	11	0.33	6.09	36.54	0.5	1	2	6	0.4	4	7	4	6	5	66.67	3	F

1.6 Dataset PC3

LocBlank	BranchCount	CallPairs	LocCodeAndComment	LocComments	ConditionCount	CyclomaticComplexity	CyclomaticDensity	DecisionCount	DecisionDensity	DesignComplexity	DesignDensity	EdgeCount	EssentialComplexity	EssentialDensity	LocExecutable	ParameterCount	HalsteadContent	HalsteadDifficulty	HalsteadEffort	HalsteadErrorEst	HalsteadLength	HalsteadLevel	HalsteadProgTime	HalsteadVolume	MaintenanceSeverity	ModifiedConditionCount	MultipleConditionCount	NodeCount	NormalizedCylomaticComplexity	NumOperands	NumOperators	NumUniqueOperands	NumUniqueOperators	NumberOfLines	PercentComments	LocTotal	Faulty
2	1	0	0	0	0	1	0.1	0	0	1	1	1	1	0	10	0	27.9	7.78	1687.93	0.07	57	0.13	93.77	217.02	1	0	0	2	0.08	28	29	9	5	13	0	10	F
1	1	4	0	0	0	1	0.07	0	0	1	1	5	1	0	14	2	40.67	14	7972.25	0.19	107	0.07	442.9	569.45	1	0	0	6	0.06	52	55	26	14	16	0	14	F
27	19	1	4	13	26	11	0.26	12	2.17	2	0.18	34	1	0	38	0	33.74	23.2	18157.82	0.26	136	0.04	1008.77	782.66	0.09	7	13	25	0.13	58	78	30	24	83	30.91	42	F
2	17	2	0	0	24	9	0.47	8	3.0	4	0.44	32	6	0.63	19	2	26.33	31.74	26522.64	0.28	154	0.03	1473.48	835.64	0.67	8	14	25	0.41	73	81	23	20	22	0	19	F
6	1	1	0	2	0	1	0.11	0	0	1	1	2	1	0	9	0	42.25	4.43	830.35	0.06	42	0.23	46.13	187.3	1	0	0	3	0.06	19	23	15	7	18	18.18	9	F
1	15	3	0	0	28	8	0.28	14	2.0	6	0.75	32	6	0.71	29	2	25.81	30.07	23331.58	0.26	143	0.03	1296.2	775.96	0.75	7	14	26	0.26	63	80	22	21	31	0	29	F
0	1	0	0	1	0	1	0.33	0	0	1	1	1	1	0	3	1	8	3	72	0.01	8	0.33	4	24	1	0	0	2	0.2	2	6	2	6	5	25	3	F
26	5	4	0	4	8	3	0.07	4	2.0	3	1	15	1	0	41	1	46.77	30.69	44054.95	0.48	243	0.03	2447.5	1435.37	0.33	2	4	14	0.04	114	129	39	21	72	8.89	41	T
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	9	0	0	0	16	5	0.45	8	2.0	1	0.2	18	5	1	11	2	8.77	18.7	3066.8	0.05	41	0.05	170.38	164	1	4	8	15	0.38	17	24	5	11	13	0	11	F

1.7 Dataset PC4

LocBlank	BranchCount	CallPairs	LocCodeAndComment	LocComments	ConditionCount	CyclomaticComplexity	CyclomaticDensity	DecisionCount	DecisionDensity	DesignComplexity	DesignDensity	EdgeCount	EssentialComplexity	EssentialDensity	LocExecutable	ParameterCount	HalsteadContent	HalsteadDifficulty	HalsteadEffort	HalsteadErrorEst	HalsteadLength	HalsteadLevel	HalsteadProgTime	HalsteadVolume	MaintenanceSeverity	ModifiedConditionCount	MultipleConditionCount	NodeCount	NormalizedCylomaticComplexity	NumOperands	NumOperators	NumUniqueOperands	NumUniqueOperators	NumberOfLines	PercentComments	LocTotal	Faulty
17	11	5	2	8	20	6	0.25	10	2	4	0.67	29	1	0	22	7	37.84	13.83	7233.63	0.17	102	0.07	401.87	523.19	0.17	5	10	25	0.11	53	49	23	12	57	31.25	24	F
2	9	3	0	1	16	5	0.56	6	2	3	0.6	17	4	0	9	1	12.5	13	2112.7	0.05	37	0.08	117.37	162.52	0.8	5	8	14	0.36	13	24	7	14	14	10	9	F
2	5	1	1	1	6	3	0.17	2	3	2	0.67	8	3	1	17	1	15.99	12.44	2476.9	0.07	44	0.08	137.61	199.04	1	2	3	7	0.13	16	28	9	14	23	10.53	18	F
4	5	1	0	0	8	3	0.3	4	2	2	0.67	11	1	0	10	0	21.06	5.85	720.66	0.04	29	0.17	40.04	123.19	0.33	2	4	10	0.19	13	16	10	9	16	0	10	F
7	5	1	3	0	0	3	0.15	0	0	1	0.33	11	1	0	17	2	21.09	13	3563.68	0.09	72	0.08	197.98	274.13	0.33	0	0	10	0.11	26	46	7	7	28	15	20	F
5	11	5	0	0	20	6	0.46	10	2	5	0.83	26	1	0	13	0	27.56	14.4	5715.21	0.13	89	0.07	317.51	396.89	0.17	5	10	22	0.25	24	65	10	12	24	0	13	F
2	1	0	0	0	0	1	0.33	0	0	1	1	1	1	0	3	0	8.98	2.5	56.15	0.01	8	0.4	3.12	22.46	1	0	0	2	0.14	2	6	2	5	7	0	3	F
2	5	0	0	0	0	3	0.33	0	0	1	0.33	9	1	0	9	1	12.72	10.31	1352.83	0.04	29	0.1	75.16	131.18	0.33	0	0	8	0.23	11	18	8	15	13	0	9	F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4	1	1	0	1	0	1	0.25	0	0	1	1	2	1	0	4	1	17.25	6.29	681.48	0.04	26	0.16	37.86	108.42	1	0	0	3	0.09	8	18	7	11	11	20	4	F

2.1 Perhitungan *Entropy Parent* dan *Entropy Children*

loc

No	loc	faulty
1	1,1	F
2	1	T
3	24	F
4	20	F
5	24	F
6	12	F
7	71	T
8	15	T

No	loc	faulty
2	1	T
1	1,1	F
6	12	F
8	15	T
4	20	F
3	24	F
5	24	F
7	71	T

Kelas	Jumlah	*pi*	Log2(*pi*)	*pi**Log2(*pi*)
T	3	0,375	-1,415037499	-0,530639062
F	5	0,625	-0,678071905	-0,423794941
Total	8	**Entropy**		**0,954434003**

No	Split	<=									Entropy <=
		T	F	Total	pi(T)	Log2(pi(T))	pi(T) * Log2(pi(T))	pi(F)	Log2(pi(F))	pi(F) * Log2(pi(F))	
1	1,05	1	0	1	1	0	0	0	-19,93156857	0	0
2	6,55	1	1	2	0,5	-1	-0,5	0,5	-1	-0,5	1
3	13,5	1	2	3	0,33333	-1,584962501	-0,528320834	0,666666667	-0,584962501	-0,389975	0,918295834
4	17,5	2	2	4	0,5	-1	-0,5	0,5	-1	-0,5	1
5	22	2	3	5	0,4	-1,321928095	-0,528771238	0,6	-0,736965594	-0,442179356	0,970950594
6	24	2	5	7	0,28571	-1,807354922	-0,516387121	0,714285714	-0,485426827	-0,346733448	0,863120569
7	47,5	2	5	7	0,28571	-1,807354922	-0,516387121	0,714285714	-0,485426827	-0,346733448	0,863120569

>									Entropy >
T	F	Total	pi(T)	Log2(pi(T))	pi(T) * Log2(pi(T))	pi(F)	Log2(pi(F))	pi(F) * Log2(pi(F))	
2	5	7	0,28571	-1,807354922	-0,516387121	0,714285714	-0,485426827	-0,346733448	0,863120569
2	4	6	0,33333	-1,584962501	-0,528320834	0,666666667	-0,584962501	-0,389975	0,918295834
2	3	5	0,4	-1,321928095	-0,528771238	0,6	-0,736965594	-0,442179356	0,970950594
1	3	4	0,25	-2	-0,5	0,75	-0,415037499	-0,311278124	0,811278124
1	2	3	0,33333	-1,584962501	-0,528320834	0,666666667	-0,584962501	-0,389975	0,918295834
1	0	1	1	0	0	0	-19,93156857	0	0
1	0	1	1	0	0	0	-19,93156857	0	0

2.2 Pencarian Nilai *Maximum Gain*

Info	Gain	Max (Gain)
0,755230497	0,199203505	0,199203505
0,938721876	0,015712127	
0,951205059	0,003228944	
0,905639062	0,048794941	
0,951205059	0,003228944	
0,755230497	0,199203505	
0,755230497	0,199203505	

2.3 Hasil *Split EBD*

Kriteria split EBD

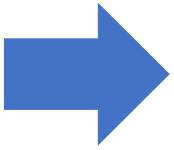
No	loc	iv(g)	e	b	branchCount
1	1,05	1	1,15	0,075	1,2
2	6,55	1,2	1185,19	0,09	2,2
3	13,5	1,7	2652,92	0,11	4
4	17,5	2	3192,33	0,125	6
5	22	2	4470,63	0,275	8
6	24	2,5	5746,48	0,71	10
7	47,5	6	19965	1,15	15

KRITERIA SPLIT TERBAIK

loc	1,05	<=1,05	min	0
		>1,05	max	1
iv(g)	6	<=6	min	0
		>6	max	1
e	1,15	<=1,15	min	0
		>1,15	max	1
b	0,125	<=0,125	min	0
		>0,125	max	1
branch Count	1,2	<=1,2	min	0
		>1,2	max	1

Dataset CM1

No	loc	iv(g)	e	b	branch Count	faulty
1	1,1	1,4	1,3	1,3	1,4	F
2	1	1	1	1	1	T
3	24	3	2936,77	0,1	9	F
4	20	2	3447,89	0,07	7	F
5	24	2	5999,58	0,12	11	F
6	12	1	2369,07	0,08	5	F
7	31	2	17846,2	0,28	7	T
8	29	3	7914,68	0,21	9	T



Hasil split

No	loc	iv(g)	e	b	branch Count	faulty
	1,05	6	1,15	0,125	1,2	
1	1	0	1	1	1	F
2	0	0	0	1	0	T
3	1	0	1	0	1	F
4	1	0	1	0	1	F
5	1	0	1	0	1	F
6	1	0	1	0	1	F
7	1	0	1	1	1	T
8	1	0	1	1	1	T

3.1 Seleksi Fitur

Dataset Hasil Diskritisasi

No	loc	iv(g)	e	b	branchCount	faulty
1	1	0	1	1	1	F
2	0	0	0	1	0	T
3	1	0	1	0	1	F
7	1	0	0	1	1	T

Kelas	Jumlah	*pi*	Log2(*pi*)	*pi**Log2(*pi*)
T	2	0,5	-1	-0,5
F	2	0,5	-1	-0,5
Total	4	*Entropy Parent*		**1**

loc	T	F	Jumlah	p(T)	Log2(p(T))	p(T) * Log2(p(T))
1	1	2	3	0,333333333	-1,5849625	-0,528320834
0	1	0	1	1	0	0
	Total		4			

p(F)	Log2(p(F))	p(F) * Log2(p(F))	Entropy	*Entropy Childern*
0,666666667	-0,5849625	-0,389975	0,918295834	0,688721876
0	-19,9315686	0	0	
		H(X) + H(Y)	0,918295834	

IG
0,311278124

Split Info	GR
0,811278124	0,383688547

SU
0,677947374

4.1 Contoh Hasil Diskritisasi yang Menyebabkan Redundansi

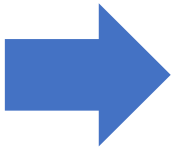
Kriteria split EBD

No	loc	iv(g)	e	b	branchCount
1	1,05	1	1,15	0,075	1,2
2	6,55	1,2	1185,19	0,09	2,2
3	13,5	1,7	2652,92	0,11	4
4	17,5	2	3192,33	0,125	6
5	22	2	4470,63	0,275	8
6	24	2,5	5746,48	0,71	10
7	47,5	6	19965	1,15	15

KRITERIA SPLIT TERBAIK				
loc	1,05	<=1,05	min	0
		>1,05	max	1
iv(g)	6	<=6	min	0
		>6	max	1
e	1,15	<=1,15	min	0
		>1,15	max	1
b	0,125	<=0,125	min	0
		>0,125	max	1
branch Count	1,2	<=1,2	min	0
		>1,2	max	1

Dataset CM1

No	loc	iv(g)	e	b	branch Count	faulty
1	1,1	1,4	1,3	1,3	1,4	F
2	1	1	1	1	1	T
3	24	3	2936,77	0,1	9	F
4	20	2	3447,89	0,07	7	F
5	24	2	5999,58	0,12	11	F
6	12	1	2369,07	0,08	5	F
7	31	2	17846,2	0,28	7	T
8	29	3	7914,68	0,21	9	T



Hasil split

No	loc	iv(g)	e	b	branch Count	faulty
	1,05	6	1,15	0,125	1,2	
1	1	0	1	1	1	F
2	0	0	0	1	0	T
3	1	0	1	0	1	F
4	1	0	1	0	1	F
5	1	0	1	0	1	F
6	1	0	1	0	1	F
7	1	0	1	1	1	T
8	1	0	1	1	1	T

4.2 Reduksi Data Hasil Diskritisasi (Biner) yang Redundan

Reduksi data redundan

No	loc	iv(g)	e	b	branch Count	faulty
	1,05	6	1,15	0,125	1,2	
1	1	0	1	1	1	F
2	0	0	0	1	0	T
3	1	0	1	0	1	F
7	1	0	1	1	1	T

= Redundansi instance dengan label yang berbeda

4.3 Proses *Split* di *EBD* Fase 2

Kriteria split EBD 2 fase

No	e		
	Fase split 1	Bining	Fase split 2
1	1,15		
2	1185,185	1	1919,0525
3	2652,92		2922,625
4	3192,33		3831,48
5	4470,63		5108,5525
6	5746,475		12855,74
7	19965,005		

Dataset CM1

No	loc	iv(g)	e	b	branch Count	faulty
1	1,1	1,4	1,3	1,3	1,4	F
2	1	1	1	1	1	T
3	24	3	2936,77	0,1	9	F
4	20	2	3447,89	0,07	7	F
5	24	2	5999,58	0,12	11	F
6	12	1	2369,07	0,08	5	F
7	31	2	17846,2	0,28	7	T
8	29	3	7914,68	0,21	9	T
9	71	9	33930,4	0,42	19	T
10	15	2	5493,37	0,13	3	T

4.4 Hasil Diskritisasi *EBD* Fase 2

No	loc	iv(g)	e		b	branch Count	faulty
	1,05	6	1,15	3831,48	0,125	1,2	
1	1	1	1	1	1	1	F
2	1	1	1	1	1	1	T
3	1	1	1	1	1	1	F
4	1	1	1	1	1	1	F
5	1	1	1	1	1	1	F
6	1	1	1	1	1	1	F
7	1	1	1	1	1	1	T
8	1	1	1	1	1	1	T
9	1	1	1	1	1	1	T
10	1	1	1	1	1	1	T

No	loc	iv(g)	e		b	branch Count	faulty
	1,05	6	1,15	3831,48	0,125	1,2	
1	1	0	1	0	1	1	F
2	0	0	0	0	1	0	T
3	1	0	1	0	0	1	F
5	1	0	1	1	0	1	F
7	1	0	1	1	1	1	T
9	1	1	1	1	1	1	T

*Sudah **tidak ada** lagi redundansi instance dengan label yang berbeda

5.1 Perhitungan *CBC*

Dataset CM1 yang sudah dikategorisasi [FOLD 1]

	No	loc	iv(g)	e	b	branchCount	faulty
	1	0	1	0	1	1	F
	2	0	1	0	1	0	T
	3	0	1	0	0	1	F
C1	4	**0**	**1**	**0**	**0**	**1**	F
	5	0	1	0	0	1	F
	6	0	1	0	0	1	F
	7	1	1	1	1	1	T
C2	8	**1**	**1**	**1**	**1**	**1**	T
	9	**1**	**1**	**1**	**1**	**1**	**T**
	10	**0**	**1**	**0**	**0**	**1**	**T**

Penentuan jumlah cluster dan titik cluster

Cluster	loc	iv(g)	e	b	branchCount
C1	0	1	0	0	1
C2	1	1	1	1	1

Jarak atribut loc terhadap titik cluster (C1 & C2)

No	loc	C1	dist
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	1	0	1
8	1	0	1

Jarak atribut iv(g) terhadap titik cluster (C1 & C2)

No	iv(g)	C1	dist
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0
6	1	1	0
7	1	1	0
8	1	1	0

Jarak atribut e terhadap titik cluster (C1 & C2)

No	e	C1	dist
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	1	0	1
8	1	0	1

5.2 Contoh Perhitungan Jarak Data Menggunakan *Euclidean Distance*

Jarak atribut b terhadap titik cluster (C1 & C2)

No	b	C1	dist
1	1	0	1
2	1	0	1
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	1	0	1
8	1	0	1

Jarak atribut branchCount terhadap titik cluster (C1 & C2)

No	branch Count	C1	dist
1	1	1	0
2	0	1	1
3	1	1	0
4	1	1	0
5	1	1	0
6	1	1	0
7	1	1	0
8	1	1	0

Jumlah jarak setiap atribut ke titik cluster

Total dist C1
1
1,414213562
0
0
0
0
1,732050808
1,732050808

5.3 Pengelompokan Anggota Data Berdasarkan Jarak Terdekat

Penentuan anggota cluster C1 dan C2

No	Total dist C1	Total dist C2	C1:C2
1	1	1,414214	C1
2	1,414214	1,732051	C1
3	0	1,732051	C1
4	0	1,732051	C1
5	0	1,732051	C1
6	0	1,732051	C1
7	1,732051	0	C2
8	1,732051	0	C2

Penentuan nilai MEAN pada setiap titik cluster (C1 & C2)

No	loc	iv(g)	e	b	branchCount
1	0	1	0	1	1
2	0	1	0	1	0
3	0	1	0	0	1
4	0	1	0	0	1
5	0	1	0	0	1
6	0	1	0	0	1
Mean	**0**	**1**	**0**	**0,333333**	**0,833333333**

5.4 Contoh Iterasi *CBC* yang Sudah Konvergen

KONVERGEN

cluster	loc	iv(g)	e	b	branchCount	faulty
C1	0	0	0	0,33333	0,833333333	F
C2	1	0	1	1	1	T

HASIL PREDIKSI

no	loc	iv(g)	e	b	branchCount	jarak ke C1	jarak ke C2	C1 : C2	faulty
9	1	1	1	1	1	1,863389981	1	C2	T
10	0	0	0	1	1	0,687184271	1,414213562	C1	F

Ground truth:

9	1	1	1	1	1	T
10	0	0	0	1	1	T

Hasil Prediksi:

9	1	1	1	1	1	T
10	0	0	0	1	1	F

6.1 Validasi

No	loc	iv(g)	e	b	branchCoun	faulty	CBC
1	0	0	0	1	1	F	F
2	0	0	0	1	0	T	F
3	0	0	0	0	1	F	F
4	0	0	0	0	1	F	F
5	0	0	0	0	1	F	F
6	0	0	0	0	1	F	F
7	1	0	1	1	1	T	T
8	1	0	1	1	1	T	T
9	1	1	1	1	1	T	T
10	0	0	0	1	1	T	F

Entropy based Discretization + Cluster Based Classification		Actual	
		T	F
Predicted	T	3	0
	F	2	5

EBD + CBC	
PD	0,6
PF	0
Balance	0,71716

BIODATA PENULIS

Fachrul Pralienka Bani Muhamad lahir di Indramayu pada tanggal 23 April 1992, anak bungsu dari dua bersaudara. Pendidikan Diploma 3 ditempuh penulis di Politeknik Negeri Indramayu, Jurusan Teknik Informatika dan lulus tahun 2012. Setelah itu, penulis melanjutkan Pendidikan ke jenjang Diploma 4 di Institut Teknologi Bandung (ITB), Sekolah Teknik Elektro dan Informatika, Jurusan Teknik Informatika dengan konsentrasi Teknologi Informasi Kesehatan dan lulus tahun 2014.

Kemudian di tahun 2015, penulis diterima di Magister Teknik Informatika, Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember (ITS), melalui jalur beasiswa BPP-DN dan berhasil menyelesaikan studi pada tahun 2017 dengan bidang keahlian Rekayasa Perangkat Lunak (RPL). Penulis memiliki ketertarikan dalam bidang RPL dan *Data Mining*. Penulis dapat dihubungi melalui alamat email fachrul.pbm@gmail.com.

[Halaman ini sengaja dikosongkan]